微计算机丛书

6809 微型机 实用技术手册

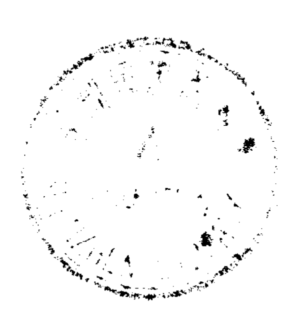
刘德贵　陆孝如　编

电子工业出版社



6809微型机实用技术手册

刘德贵　陆孝如　编



電子工業出版社



内 容 简 介

本书详细说明美国莫托罗拉公司6809高性能八位微型机的基本工作原理、硬软件设计特点以及系统组成和应用。全书共分五章，第一章介绍6809的基本特点；第二章详细说明6809硬件结构、电气性能、内部结构和外部引线；第三章详细研究6809软件设计、指令系统、寻址方式和现代程序设计方法；第四章说明6809接口方式和系统组成及应用，基本接口和MC6809存储器管理器件的虚拟存储原理以及一般应用系统；第五章详细给出6809某些实用源程序文本和使用说明。

本书编有20个附录，为实际应用6809提供了较详细的参考数据、图表、程序和资料，以及150个问答习题。

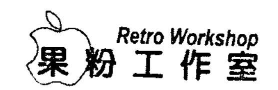
书末给出参考文献资料。

本书可供从事计算机工程和电子工程等方面的工程技术人员和大专院校有关专业师生参考。

6809微型机实用技术手册
刘德贵　陆孝如　编
责任编辑　邓 云 溪
*
电子工业出版社出版（北京市万寿路）
北京市昌平环球科技印刷厂印刷
新华书店北京发行所发行　　各地新华书店经售
*
开本：787×1092 1/16　印张：28.375　字数：655千字
1986年7月第1版　1986年9月第1次印刷
印数：4.300　　定价：5.50元
统一书号：15290·421



目　　录



第一章　有关6809的概述

1.1　6809的特点

6809的最大特点就在于为提高软件的开发效率而增加了寄存器，并强化了指令系统和寻址方式。

1. 6809所处的地位

6809是由于超大规模集成电路技术（VLSI）的发展而出现的一种微处理机，它是68系列的较高挡机种。图1.1中给出了集成度、处理能力的比较关系，6809位于中央，其集成度和处理能力都约为6800的 4 倍。6809不象6801的发展方向那样，把ROM、RAM、TIMER、PIA等全都集成在一片电路之内，它的发展方向是采用比较高速的硬件，简化软件的方法来改进处理能力。

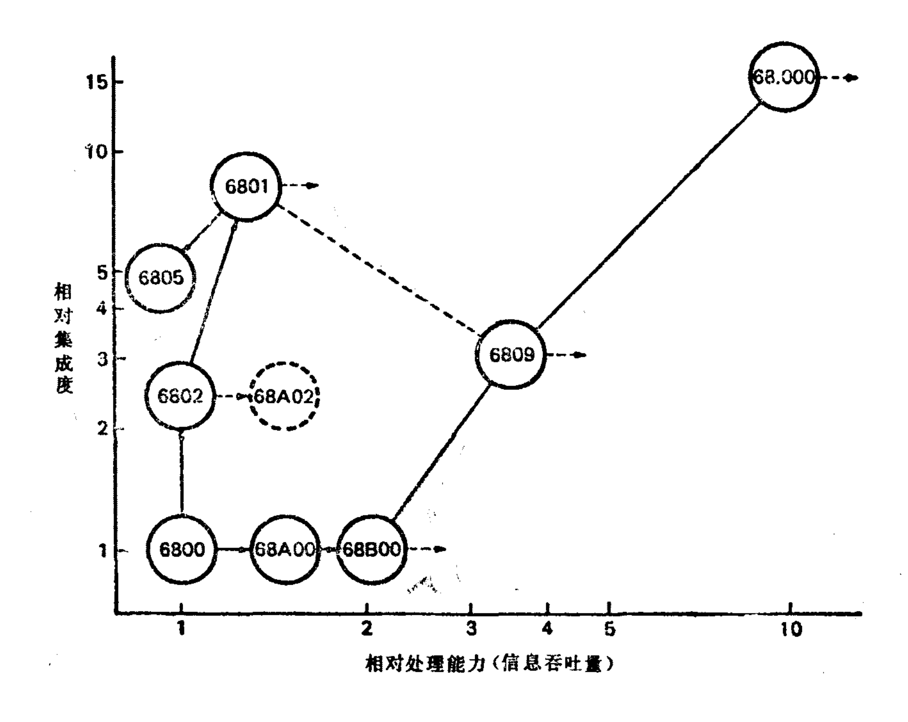


图1.1　莫托罗拉公司微处理器集成度和处理能力的比较

在深入研究6809之前，根据莫托罗拉公司提供的资料可以看到6809同6800及其它微处理器的比较情况。图1.2是各种处理器相对执行时间的比较，从中可以看出6809的性能接近16位机。表1.1、表1.2、表1.3分别说明各种微处理器执行 8 种基本测试的相对执行时间和实际执行时间，以及各处理器综合性能的比较结果。从这些资料中可以说明6809是 8 位机中最好的一种微处理器。

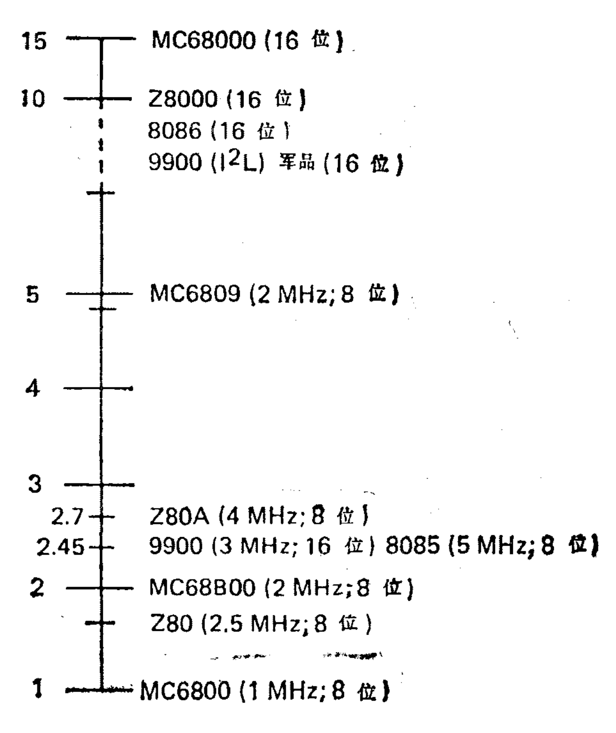


图1.2　微处理器相对执行时间比较

2．　6809硬件兼容6800

6809硬件和6800完全兼容，因此全部6800系列的外围器件均可在6809中原样照用。

6809的总线中有E脉冲线，可以采用接近方波的同步信号，使外围器件和存储器完全在同步状态下工作。由于具有这种工作方式，所以很容易进行同步定时的设计。

6809、68B09的E脉冲信号的标准频率分别为1 MHz和2 MHz。不必象Z80或8080系统中在周期之半处工作时，需要插入中断处理等待时间（WAIT周期）。

此外，6809除具有6800二级硬件中断$\overline{\text{NMI}}$和$\overline{\text{IRQ}}$外，又增加了一级快速中断$\overline{\text{FIRQ}}$，共有三级硬件中断。

6809芯片中设有时钟电路，不象6800那样需外接时钟。

6809是8位微处理器中唯一可接MMU（存储器管理单元）的器件。MMU可使64K字节的存储空间得以扩充，并能把内存按片分配给多个程序使用。

存储管理本是大、中型计算机所采用的一种技术，在8位机上能实现这种技术，表明微计算机技术在迅速发展。

3．　软件向上兼容

6809与6800的汇编源程序向上兼容。把6800用的源程序放在6809的系统上运行时，必须注意以下几点：

在6800中，没有根据接在指令（操作码）后的修改字节（称后缀字节——POSTBYTE）来指定寄存器的方式，而在6809中，有许多指令是根据后缀字节指定寄存器的。因此，将表3.23所示的记忆符改变不会有什么问题。如把6800的源程序放在6809的汇编程序中，按照表3.23的指令对应关系能全部自动地变换为6809的汇编程序，不必要修改记忆符。但是，当程

表 1.1　8 种基本测试的相对执行时间

		I/O 处理程序	字符检索	计算转移	双字长右移 5 位	16位向量加	8 位向量加	16×16位乘　法	数据块传送(64字节)	平　均执行时间
6809	2.0MHz	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	1.5MHz	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
	1.0MHz	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
Z80	4.0MHz	1.4	0.8	2.1	2.7	1.6	1.8	3.3	1.0	1.8
	2.5MHz	2.2	1.2	3.4	4.4	2.6	2.9	5.2	1.6	2.9
9900	3.0MHz	2.6	2.3	2.8	1.5	1.7	3.0	0.5	1.6	2.0
6800	2.0MHz	0.9	1.4	1.9	1.3	3.1	2.8	5.0	3.3	2.4
	1.5MHz	1.2	1.9	2.5	1.7	4.1	3.7	6.7	4.3	3.3
	1.0MHz	1.8	2.8	3.7	2.5	6.1	5.5	10	6.5	4.9
8080+	3.0MHz	1.9	1.8	2.8	6.1	2.3	2.7	9.6	2.4	3.7
8085	2.0MHz	2.8	2.6	4.2	9.1	3.4	4.1	14.3	3.7	5.5

表 1.2　8 种基本测试的实际执行时间(μs)

		I/O 处理程序	字符检索	计算转移	双字长右移 5 位	16位向量加	8　位向量加	16×16位乘　法	数据块传送(64字节)	平　均执行时间
6809	2.0MHz	28	287.5	34.5	15	325	180	82	344.5	
	1.5MHz	37.3	383	46	20	433	240	109.3	459.3	
	1.0MHz	56	575	69	30	650	360	164	689	
Z80	4.0MHz	38.3	220.5	73.3	41	518	323	267	342	
	2.0MHz	61.3	352.8	117.2	65.6	828.8	516.8	427.2	547.6	
9000	3.0MHz	72	661	98	22	537	537	42	537	
6800	2.0MHz	24.5	404	64.5	19	993.5	498.5	409.5	1123.5	
	1.5MHz	32.7	539	86	25.3	1325	665	546	1498	
	1.0MHz	49	808	129	38	1987	997	819	2247	
8080+	3.0MHz	52.7	506.7	96.7	91.3	732	492	784	841	
8085	2.0MHz	79	760	145	137	1098	738	1176	1262	

表 1.3　处理器综合性能比较

性　能　指　标	MC6809	Z-80A	MC6800	8085
指令数目	1.0	1.50	1.72	2.30
字节数目	1.0	1.31	1.58	1.80
微秒数目	1.0	1.80	2.40	2.20
	(2MHz)	(4MH2)	(2MHz)	(5MHz)

设6809标准值为1.00

序的进入地址和程序的字节数有变动、或者分支转移指令溢出时，则需要修改一部分源程序。

6809条件码寄存器中的高二位不象6800那样均为1，而是作为E（输入标志位）、F(FI-RQ屏蔽位）位使用的，所以在程序区中使用条件码时，这一点要特别注意。

4. 软件技巧的改进

6809为充分发挥程序设计技术提供了极其优越的性能。6809不完全是对6800直接寻址方式的改进，而是可以应用位置独立程序设计（也称自由地址程序设计）、结构程序设计、可再入和循环调用程序设计等最新的程序设计方法进行程序设计。6809本身充分支持了这些程序设计方法，为其应用开辟了广阔的前途。

1.2 6809的体系结构

1.2.1 累加器和寄存器

6809有两个8位的累加器，两个可以变址的通用16位寄存器，两个可以变址的堆栈指示器。由于设有直接页面寄存器，因此可以把直接页面区置于64K字节的地址空间中的任意地址之内。

1. 累加器（A、B、D）

6809设有两个8位的累加器ACCA和ACCB，这两个累加器除在下述特殊情况下都具有相同的功能。

一般讲，累加器是执行加法、减法、乘法移位等运算时所必须的一种通用寄存器。6809中可使ACCA和ACCB进行双字长运算，此时令ACCA为高位字节、ACCB为低位字节串联成一个累加器进行工作。串联之后称为ACCD，记忆符也用ACCD表示。因此可以执行16位算术逻辑操作和传送、交换操作。

累加器A还有特殊用途，即在做完十进制加法之后，可以执行十进制数的调整（执行DAA指令）。累加器B不能执行这种指令。只有累加器B才能执行的指令是ABX（X←X+B）。

2. 变址寄存器（X、Y）

MC6809设有两个16位的变址寄存器X、Y，它们可在64K存储器空间进行变址修改。所谓变址，就是根据指令所给出的偏移量和指针寄存器（X、Y、U、S，有时还有PC）的内容，算出有效地址。

6809中的两个变址寄存器也可以作为通用寄存器来使用。变址寄存器按其本来的功能相当于书籍中的目录，可以表示数据的起始位置，或表示处理过程中数据所在的位置。

3. 堆栈指示器（U、S）

6809设有可以作为变址寄存器使用的两个16位的堆栈指示器U和S。6800、8080、Z80等只设有一个堆栈指示器，但6809设有硬件（系统）堆栈指示器S和用户堆栈指示器U。堆栈指示器的工作原理顾名思意的理解就是作为堆栈用的指示器，它对堆栈区进行管理。所谓堆栈就是指保留累加器或寄存器内容用的存储区。堆栈指示器S的作用就是将子程序的返回地址或中断发生时的全部或部分累加器和寄存器的内容，自动地保留在系统的堆栈之中。同样，用户堆栈指示器U可在用户程序需要时把累加器和寄存器的内容也暂时保留在堆栈之中，因

此无论系统堆栈还是用户堆栈它们都可以根据堆栈指示器自动地进行更新。但在进行这些处理之前，需要把各堆栈区的最后地址在一开始就预先设置在各个堆栈指示器之中。

U和S也可以和X、Y一样用作变址寄存器。

关于用户堆栈指示器U的使用方法还将在1.2.2新设寄存器的效用一节中进行说明。

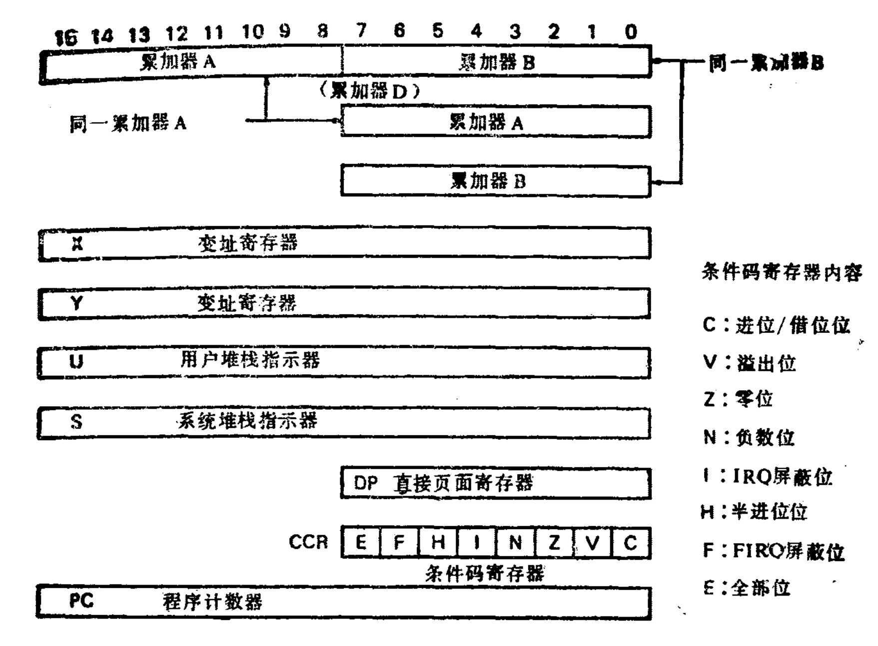


图1.3　6809寄存器的组成

4. 直接页面寄存器（DPR）

6800寻址方式中有直接寻址方式，在存取直接页面的情况下，一条指令可以节约存储器一个字节，速度也可以快点。6809设置直接页面寄存器实现直接寻址方式时，就不限于6800那样只有256字节的地址范围，而是可以访问全部地址空间，直接页面寄存器是为实现此功能所新设置的寄存器。

采用直接页面处理方式是68处理器系列中很大的一个特点，它把大量的通用寄存器不是放在CPU芯片内，而是移于外部存储器中，这是一种概念上的突破。

6800中可作为直接页面使用的地址设在＄0000到＄00FF之间。而6809是把8位字长的直接页面寄存器的内容作为直接寻址方式中的高位地址使用，所以6809可以把全部64K字节的地址空间作为外部通用寄存器使用。

直接寻址方式是缩短程序，提高运算速度的一种有效的手段。DPR的内容只要一次设定，不必改写，就能随便访问该页内的任一单元。在系统复位时DPR的内容由$\overline{\text{RESET}}$信号清零，这就保证了与6800的兼容性。

5. 条件码寄存器（CCR）

条件码寄存器是根据处理器内部执行过程中所产生的状态进行记忆的寄存器。

在6800的CCR中，最高二位没有使用，而且都设置为“1”，但在6809中全部数字位都用上了。从最高位开始各位的顺序是E（全部标志位——Entire flag），F（快速中断屏蔽位——Fast interrupt mask），H（半进位位——Half carry），I（中断屏蔽位——

Interrupt mask，N(负数位——Negative)，Z(零位——Zero)，V(溢出位——Overflow)，C（进位位——Carry）。

有关各位的含意见第二章2.2节。

6. 程序计数器（PC）

程序计数器是管理程序流的16位计数器，6809的程序计数器与6800、8080、Z80所具有的概念完全不同。它有以下两个特点：

（1）可作为变址寄存器工作

在按相对寻址方式编制位置独立程序的过程中，凡执行数据输入输出工作的时候都可以利用程序计数器进行。详细情况在“3.2.9偏值的给定”一节中将加以说明。

（2）具有和其它16位字长的寄存器进行交换传送的功能

在利用累加器D或其它16位寄存器的运算中，或者在进行表格访问而引起的程序流的变动，即所谓计算机中的GO TO（转移）处理等过程中，具有最适宜的功能。

在执行程序语句中间，分离输入输出程序的方法是调用例行程序，这时就可以用寄存器和程序计数器进行交换的方法来完成。

1.2.2 新设寄存器的效用

1. 累加器D

累加器D是累加器A做高位、累加器B做低位组成的双倍长累加器。以累加器D为对象的指令有双倍长加法（ADDD）、双倍长比较（CMPD）、双倍长装入（LDD）、双倍长存储（STD）以及双倍长减法（SUBD）等五种。

在6809汇编程序中，对于三种移位指令（ASLD、LSLD、LSRD）、压入弹出指令也可以使用累加器D编写。

在8位数据和16位（双倍长）数据间的加减法中，如ACCB＝$FF(－1)且ACCA＝0，这时ACCD＝$OOFF不能表示－1。为了表示－1，必须使ACCD＝$FFFF。因此对ACCA的处理应用SEX指令来完成。即在使8位数据变换为16位数据时需要用SEX指令进行处理。

累加器D和其它的16位寄存器X、Y、U、S或PC之间的交换（EXG）或者传送（TFR）都是可以的。

2. 变址寄存器X、Y

两个变址寄存器X、Y，如果除去ABX指令以外，几乎具有同样的功能。从指令编码表中可以看出：变址寄存器Y的指定，可以根据变址寄存器X的指定进行页面转换得到。例如CMPX＝$9C、CMPY＝$109C，LDX＝$9E、LDY＝109E，由此可见对变址寄存器Y的指定只是付加了先行字节（前置字节）$10。

对于LEA、TFR、EXG等指令的后缀字节来说，在指令执行过程中完全相同。特别在位置独立型程序中使用LEAY DATA PCR等程序计数器相对寻址的指令时完全没有浪费。

变址寄存器X作为在LDX、STX等数据传送指令中使用的变址寄存器，变址寄存器Y作为在位置独立型程序和程序内常数管理中使用的变址寄存器，这样对程序缩短，速度提高也会有所改善。

只在变址寄存器X中才可以执行ABX指令，这对255字节以内的数据块的访问是很有好处的。（参照表1.1基准测试的结果）。

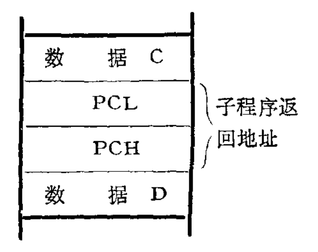
3. 用户堆栈指示器

用户堆栈指示器U和X、Y一样可以作为变址寄存器使用，不言而喻，全部寄存器也都可以保留在用户堆栈区之中。但是用户堆栈指示器有以下完全不同的用法。

（1）符号序列（数据序列）的变换

利用用户堆栈输入五个数据A、B、C、D、E，要求输出时数据顺序为A、C、E、D、B时，需要按以下情况处理。

输入	执行	输出
A	按原内容输出	A
B	保留到用户堆栈	
C	按原内容输出	C
D	保留到用户堆栈	
E	按原内容输出	E
	从用户堆栈返回	D
	从用户堆栈返回	B



如果上述这种处理在系统堆栈内进行时，因为要转移到子程序进行边输入边输出，所以要往系统堆栈内保留子程序返回的地址，因而系统堆栈指示器就不能正确表示数据序列的位置。在本例中为了输入数据D，如果利用子程序进行，那么在系统堆栈之内就不能插入子程序返回地址。

（2）波兰表示法的处理①

处理数值计算公式时，人们擅长按数学算式的形式输入，可是对人们一看就能理解的计算顺序，而计算机却不能直接执行。因此需要把数学形式的输入改写为计算机所擅长的波兰表示法。

例如处理A＊(B＋C/D)时，从左边开始读入算术式中每个字符，由于要求被运算数A、B、C、D的顺序不改变地写出，因此就要变动运算符号的顺序和相对位置。运算符号的优先顺序为＊＊（↑），＊或/，＋或－。如果有符号“(”出现，则优先顺序暂时冻结，当出现符号“)”时，则在此以后解除冻结。求解方法如图1.4所示，结果为ABCD/＋＊。图1.5为执行ABCD/＋＊实例。

在图1.4中写有K的方框，相当于6809的用户堆栈指示器。

为了处理算术式句法的分析，以及按波兰表示法变换为执行语句时，用户堆栈指示器是必不可少的。特别是对PASCAL、LISP等语言使用再入性程序设计实现结构化程序时，为了执行所写的程序，这种用户堆栈功能也是不可缺少的。

4. 直接页面寄存器DPR

6800中有直接页面寻址方式，但它是把16位地址空间的低8位地址规定为指令的操作数，而高8位地址字节在6800中为$00；在6809中则给出直接页面的数据。

6809的直接页面寄存器的内容为了保持和6800之间兼容，可用 $\overline{\text{RESET}}$ 使其复位为$00。

① 这是一种无括号的表示法，可用来写逻辑表达式、算术表达式和代数表达式。这种方法是波兰逻辑学家J.Lukasiewicz首先提出来的。故被命名为波兰表示法。

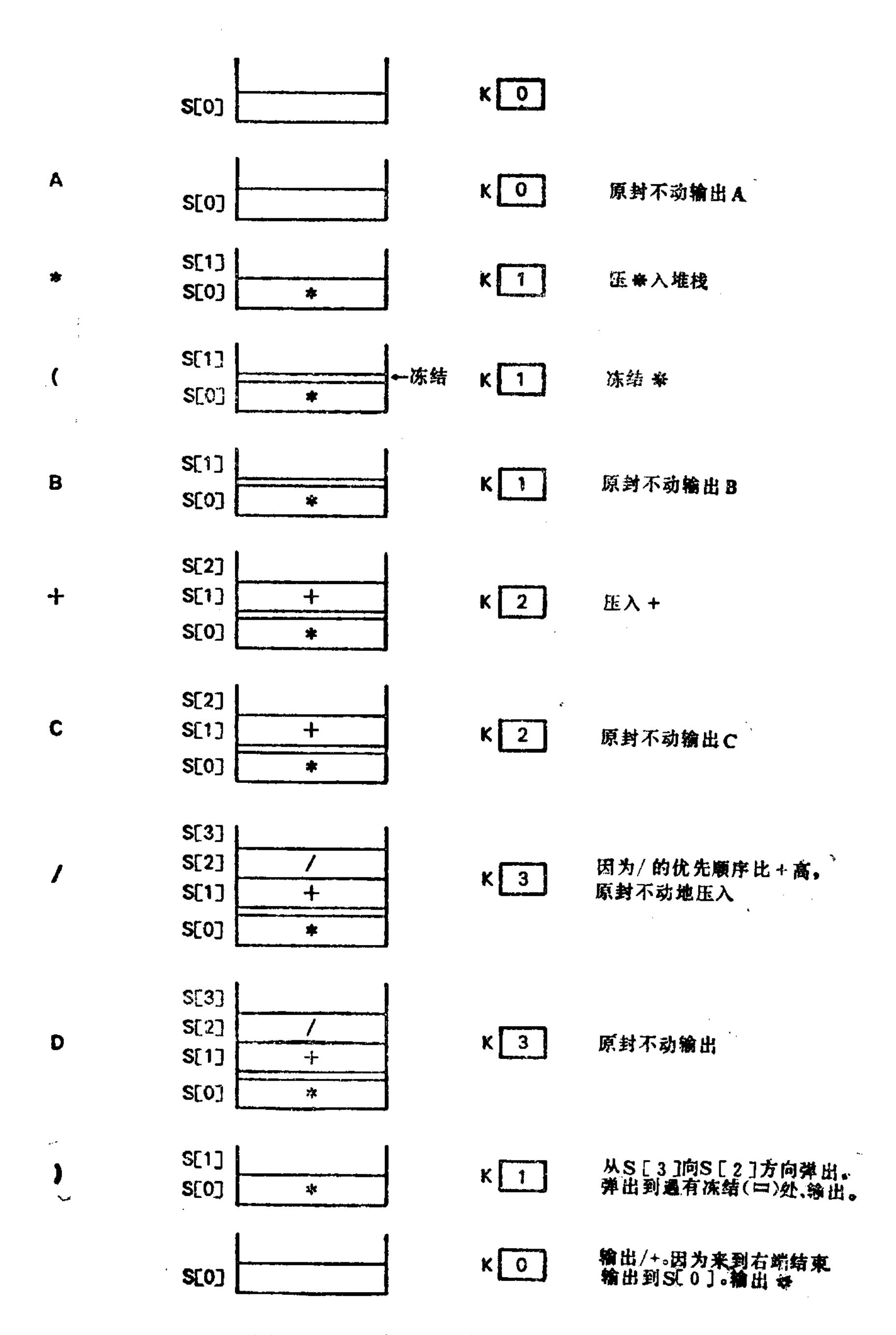


图1.4 A＊（B+C/D）的波兰表示法

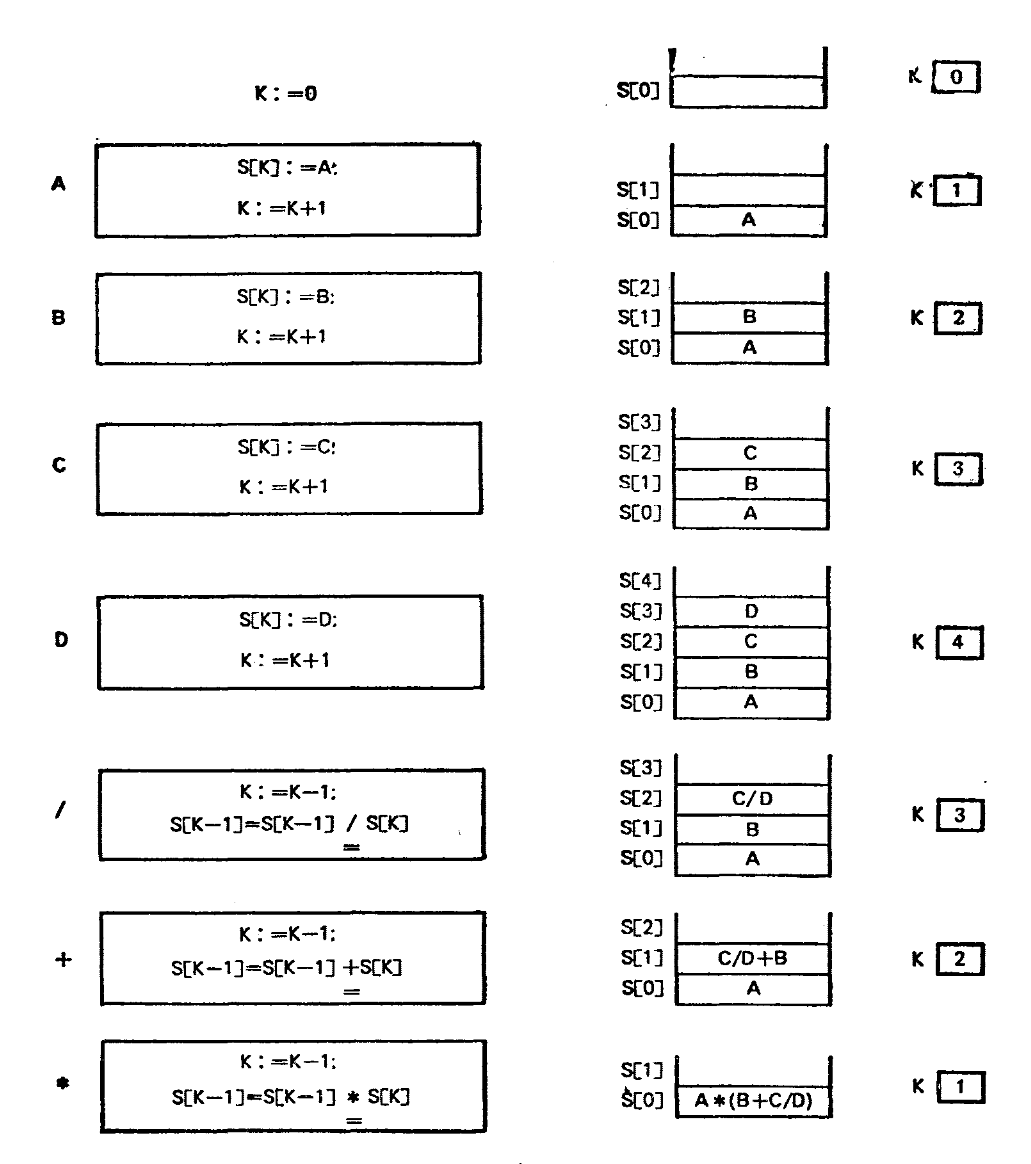


图1.5　ABCD/+*的执行实例

对直接页面寄存器没有设置直接装入的指令，需要先装入到累加器A或累加器B之后，再用TFR指令传送到DPR之中。即

```
LDA    #$FF      ACCA←$FF
TFR    A,DP      DP ←(A)
```

本例装入DPR的内容为$FF，则直接页面区为$FF00～$FFFF。

把被装入的起始地址作为直接页面方法的程序实例写在5.2.1节8080仿真程序之中。当利用存储器管理部件时，不管在哪个主存储器上运行位置独立程序，用直接页面法去进行存储器分配都方便易行。

在图1.6设置直接页面实例中，基本页面（直接页面）为256字节，程序区为512字节、程序区接在数据区之上。

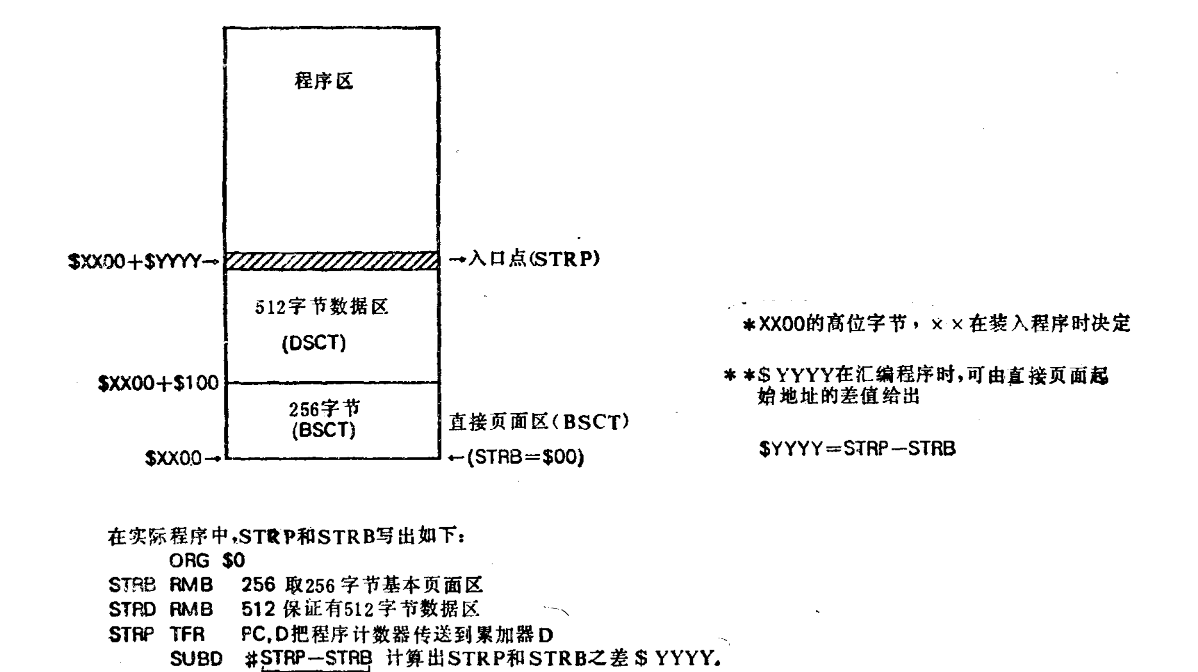


图1.6 直接页面寄存器设置实例

如果把该程序的入口作为程序的起点，STRP和STRB之差为$300字节。现在该程序从$6300地址被装入。这时如果从STRP开始执行，当把PC的数值送到累加器D时，累加器的内容为$6302，表示为下条指令的操作码地址。如果执行$6302－$300，累加器D的数值即为$6002，再把累加器A的数值$60装入直接页面寄存器。经这一连串过程执行之后，直接页面寄存器内容为$60，直接页面区从$6000到$60FF。

这样，如果根据程序计数器的内容求出直接页面寄存器的数值，就可在位置独立或再入的程序中使用直接寻址方式进行工作。

1.3 6809的软件特点

设计6809时，尤为突出地考虑了提高软件使用性能，这是比其他8位微处理器机种优越的重要方面。

1.3.1 寻址方式和指令系统

1. 6809有10种寻址方式

（1）固有寻址（也包括累加器寻址）；

（2）立即寻址；

（3）扩充寻址；

（4）间接扩充寻址；

（5）直接寻址；
（6）寄存器寻址；
（7）变址寻址；
（8）间接变址寻址；
（9）相对寻址；
（10）程序计数器相对寻址。
另外变址寻址、间接变址寻址还有四种方式：
（1）零偏移值；
（2）常数偏移值；
（3）累加器偏移值；
（4）自动加1或减1。
有关这些寻址方式的工作原理，将在3.2寻址方式一节中详细说明。

2. 指令系统

6809的基本指令包括乘法(MUL)、16位运算（LDD、STD、ADDD、SUBD、CMPD）和全寄存器间的传送交换指令在内共有59条指令。

这59条基本指令，用汇编语言编写出的指令当它们被翻译为机器字编码时可有1464条指令。如若这1464条指令象8080、Z80那样到现在还没有整理出适于记忆的表达形式，那么程序人员必须背记1464个记忆符。这样，煞费苦心设计的6809也就谁都不能使用。然而用59个记忆符可以表示1464个机器字，这就是6809功能很强，而又易于使用的理由。

6809在软件设计上，既考虑要达到兼容6800软件，保持6800系统的能力，但又不要使其限制在6800机器码定义的指令上，因而选择在汇编语言的源码指令定义上兼容6800指令。因此，为6800准备的任何汇编语言的程序都可以用6809汇编程序通过后产生在6809机器上运行的目的代码。

6809所增加的最有效指令性能如下：

（1）使用寻址系统的能力来取代6800变址寻址方式。寻址系统的能力有以下方面：它可以用6809四个变址寄存器和堆栈指示器中的任何一个寄存器给出操作数或操作数的地址（间接寻址）；指示器可以是固定的或可变的带符号的偏移值；指示器可以选为自动加1或减1的工作方式；程序计数器可以用来作为指示器而存取操作数或相对寻址时的操作数地址；最后，还可用任何一种变址寻址方式来形成任何变址寄存器或堆栈指示器中的地址。

（2）A和B两个累加器可以构成一个16位D累加器并可执行16位的加、减、比较、装入和存储指令的操作；还可把A、B或D累加器的内容加到CPU中任何一个变址寄存器和堆栈指示器之中；而且还可在CPU寄存器中使任何长度相同的寄存器之间完成内容相互交换的任务。所以6809中指示器的处理能力有了极大的提高。

（3）具有乘法指令，可在A和B累加器中实现两个8位无符号数的乘法。

（4）凡两字节的指令都可以对任何一个堆栈同CPU寄存器之间执行任意次数的压入或弹出。

（5）可用相对寻址的“长分支转移”指令寻址到全部程序的地址。

（6）6809新设置一条同步指令，用来提供软件同外部硬件过程中的同步。此时CPU保持在停止状态，等待接收外部中断后再运行。接收中断后CPU即处理中断，当从中断返

回时清除停止状态，继续按指令顺序执行。

1.3.2 软件结构控制

1. 位置独立型程序设计

所谓位置独立程序，就是指某个程序不加任何改动地放在存储器的任何地址上都能正确进行工作的程序。

对于位置独立程序，当使用其内部常数等数据或作业区时，当然不能使用绝对寻址，使用直接寻址时也要注意是否可行。但可以使用固有寻址、立即寻址、变址寻址或相对寻址等方式。在6809中由于具有相对寻址的条件分支转移和无条件分支转移，相对寻址的子程序调用（分支转移到子程序），程序计数器相对寻址、寄存器寻址、以及选用丰富的变址寻址等方法，不使用绝对寻址的程序很容易实现。因此，使用位置独立程序后，象从磁盘到RAM准备程序时就不需要用“再定位装入程序”即可把程序定位在任何地址上工作。

2. 再入程序和循环调用程序设计

所谓再入程序是指几个不同用户或几个任务共用一个程序。由于共用一个程序可以节约存储空间，这一点在中断处理的时候特别重要。例如，对某个程序来说，NMI程序和IRQ程序都同时调用它时，就会相互破坏对方的中间结果，但如果改变了程序的流向，就会得到完全不受影响的程序。

6809为了编出再入程序，大致有两种方法。第一是利用堆栈内的数据处理。6809的堆栈本身的数据处理能力是很强的，6809可提供四个堆栈的使用能力，如果把两个堆栈指示器都作为变址寄存器使用时，那么再入程序就会很容易地编制出来。所以使用堆栈编制再入程序的方法非常简便。

第二就是利用直接页面寄存器的方法。例如，可把NMI进行处理的直接页面寄存器的内容设为$ EO，IRQ进行处理的设为$ E1，这样相互有关的数据区、和暂存的中间结果就不致被破坏。但用第二种方法时就需要在各自处理过程中，每次都要对直接页面寄存器置位。

循环调用程序是指可以进行自身调用的程序，这在语句的句法分析程序中以及计算阶乘等各种函数运算程序中极为简便易行。和再入程序一样使用堆栈方法甚为方便。

3. 模块化结构程序设计

所谓模块化程序，就是把编制出的程序作为“程序零部件”。它们可以在其它系统中再使用，或者把系统中某个部分换成其它“程序零部件”，或从该系统取出某个“程序零部件”。这些“程序零部件”都可称为程序模块，简称模块。6809支持了复杂的高级的模块结构式语言，如象需要强大堆栈处理能力的PASCAL语言。由于6809设有16位堆栈指示器和四个独立的堆栈，因此对于模块化结构的PASCAL语言特别合适。PASCAL语言本身即具有可再入的程序结构，不使用跳越转移指令，而是利用组合统一模块的程序，把各个模块程序一个个结构而成。因此，把6809的模块程序（子程序）如用堆栈指示器编成处理程序极易于实现。在开始执行模块时，首先6809用一条指令（PSHS Y，X，B，A）把寄存器的内容压入堆栈，当结束执行模块时，可以用另一条指令（PULS A，B，X，Y，PC）来恢复被保留的寄存器内容，此时不用RTS指令就可以使PC弹出堆栈。因此6809允许按高级模块化结构语言Pascal来书写程序，而且可被编译为最有效的、运行最快的机器码，因为6809可以直接处理模块化结构的高级语言。

另外，6809的每个模块或子程序，需要在系统堆栈之内保证该模块或子程序所需要的缓冲区（局部变量）时，用LEAS指令可方便地实现。

同时，在几个模块或子程序之中，需要使用公用数据（全变量）时，可以利用用户堆栈指示器来进行。因此具有二个堆栈指示器的6809可以实现结构化数据。在系统堆栈区内被保留的局部变量在执行结束时就同时被取消，因此提高了存储器的利用率。

如果把程序中要使用的某个模块或子程序存在一页表格之中，那么6809从表格中读出程序或调试程序时是非常方便的。6809易于编成读出的表格，也是为程序人员解决了一项重要工作。

微计算机系统设计人员使用6809进行软件开发时，允许把所写的程序象ROM那样，按通用方式进行分配，可规定在任意地址之上，而且不需同任何的其它软件进行接口。因此，用6809开发的任何应用软件包都可以作为软件包ROM，而不会产生象其它机种软件包所受的限制和缺点。6809设计的程序可按模块方式写出，各部分的相互联系均按结构程序设计的顺序进行工作。

4. Pascal程序实例

现在我们以Pascal实际程序之例来说明6809编程序的优点。

表1.4给出的是下标数组的Pascal程序摘录。采用普通的6800或8080来编译这种结构的

表1.4 下标数组的Pascal程序摘录

```
TYPE
        index=0…10;
        twiceindex=0…20;
        unsigned=0…32767;
        short=-128…127;
        short unsigned=0…255;
        thing=record
                    field1: 0…7;
                    field2: 0…31;
                 end;
        packed Thing=packed record
                    field1:0…7;
                    field2:0…31;
                 end;
VAR
        a,b:array[index]of integer;
        i,j:index;
        k:twiceindex;
        s:set of(READY,BLOCKED,RUNNING,SWAPIN,SWAPOUT);
BEGIN
        a[i]:=b[j]; {the dreaded array-indexing example}
        k:=i+j;              {subranges are useful}
        s:=[READY,BLOCXED,RUNNING]; {set operations}
        s:=s-[READY,RUNNING];
        s:=s+[SWAPIN];
        s:=s*[SWAPIN,BLOCKED];
END
```

程序时（甚至是最简单的算术表达式或指示字表示式），其目标码和源码之比很高。表1.5所给的内容就是用6800来实现表1.4中Pascal赋值语句a〔i〕:=b〔j〕的编译程序。这里假设自动数组用属于某处存储区的堆栈指示字来实现。另外还假定编译程序要考虑其自动变量对堆栈指示字移动的堆栈位移值。这里采用j表示变量j的堆栈位移值。另外，在这段程序中，对过程的起始段必需设置数组a和b的指示字（存放的位移值分别为a和b），以便在数组开始执行之前确定在整数上。例如a〔1〕即被认为是数组a之首项。

给数组赋值用6800处理是很冗长的，同样处理采用6809则是较简便的，如表1.6所示；为了进一步比较说明，采用PDP-11/45处理的汇编程序如表1.7所示。

表 1.5　表 1.4 Pacscal 程序中第一行用6800编译的汇编程序

```
TSX                 / Enable indexing off SP
LDA   A,j(x)        / Fetch address of j relative..
LDA   B,j+1(x)      / to sp into (A,B) register pair
ASL   B             / shift (A B).pair left by 1..
ROL   A             / yielding integer offset
ADD   B,b+1(x)      / Add in 16-bit array
ADC   A,b(x)        / pointer i to (A,B) pair
STA   A,temp        / Transfer (A,B) pair to reg..
STA   B,temp+1      /..not re-entrant
LDX   temp
LDA   A,o(x)        / Finally, fetch b [j] into..
LDA   B,1(x)        / (A,B) pair..
PSH   A             / and push onto stack
PSH   B
TSX                 / Following code is repeat of..
LDA   A,i(x)        / above for getting address of..
LDA   B,i+1(x)      / array element a[i]
ASL   B
ROL   A
ADD   B,a+1(x)
ADC   A,a(x)
STA   A,temp
STA   B,temp+1
LDX   temp          / x now points at a[i]
PUL   B             / pop b[j] from stack..
PUL   A             / into (A,B) pair..
STA   A,o(x)        / and store in a[i]
STA   B,1(x)
Total code:52 bytes
```

虽然以下标数组为例可以说明机器寻址方式的灵活性，而且赋值语句也可以检验寄存器的各种用途。只以一个例子推论，可能是不够的，不过美国瓦特鲁大学的C.H.FORSYTH和R.T.HOWARD二人对更一般的算术式和函数以及过程调用、保留、和返回序列都做了深入的比较，经过他们的试验，认为以下标数组来说明6809的特点是具有典型性的。

综上所述，6809软件中许多强大功能都表现在变址寻址方式的灵活性上，对汇编语言的用户来说，使用6809所发挥功能的强弱取决于对这些寻址方式和LEA指令使用总数的多少。而对高级语言用户来说，将会发现这些功能极强。

表1.6　下标数组的6809汇编程序

```
/ 'x' points to top of stack  (display)
LDA    D,i(x)          / i
ASL    B
ROL    A               / *2
ADD    D,$a-2          / +offset of 'a'
LEA    Y,D(x)          / +stack top
LDA    D,j(x)          / j
ASL    B
ROL    A               / *2
ADD    D,$b-2          / +offser of'b'
LDA    D,D(x)          / +stack top
STA    D,(Y)           / a[i]:=b[j]
Total code:20 bytes
```

表1.7　下标数组的PDP-11汇编程序

```
/ r5 points to the "top" of the
/ stack frame
MOV    j(r5),ro          / j
ASL    ro                /*2
ADD    r5,ro             /+display pointer
MOV    i(r5),r1          / i
ASL    r1                /*2
ADD    r5,r1             /+display pointer
MOV    b-2(ro),a-2(r1)   / a[i]:=b[j]j
Total code:22 bytes
```

1.4　6809的硬件特点

6809除了在体系结构和软件性能上作了改进之外，在硬件上也作了许多改进。其中最重大的改进有中断、控制信号以及有关的控制线的功能。6809的单片集成度每片大约有15000个晶体管，为6800的三倍。因此在硬件结构上有了明显的改进和提高。

6809和6800一样有40条引线，做成塑料封装（带P字）和陶瓷封装（带L字）两种器件。其总线和控制信号线如图1.7所示。6809芯片有两种结构形式：一种是MC6809，一种是MC6809E。前一种内含晶体振荡电路，后者从外部加上时钟信号才能工作。6809E的总线和控制信号线如图1.8所示。

6809在硬件改进上有以下内容：

（1）时钟振荡器设计在芯片之内，不需要单独时钟器件，只要求有一晶体或时钟作为脉冲源即可。其晶体频率要求为4倍的内部时钟频率。

（2）设有处理器挂起的状态输入线DMA/BREQ，可为DMA直接存取和动态存储器更新等其它应用请求CPU的总线进行工作。

（3）增加了快速中断请求输入，中断时保留在堆栈中的CPU寄存器只有程序计数器和条件码寄存器。6809设有三级中断，高级中断可自动禁止低级中断的工作。

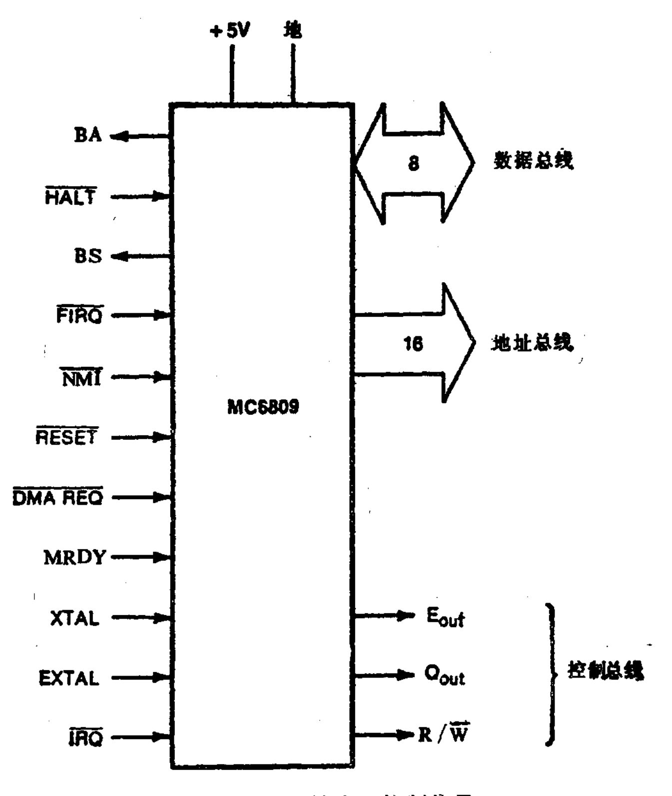


图1.7 6809总线和控制信号

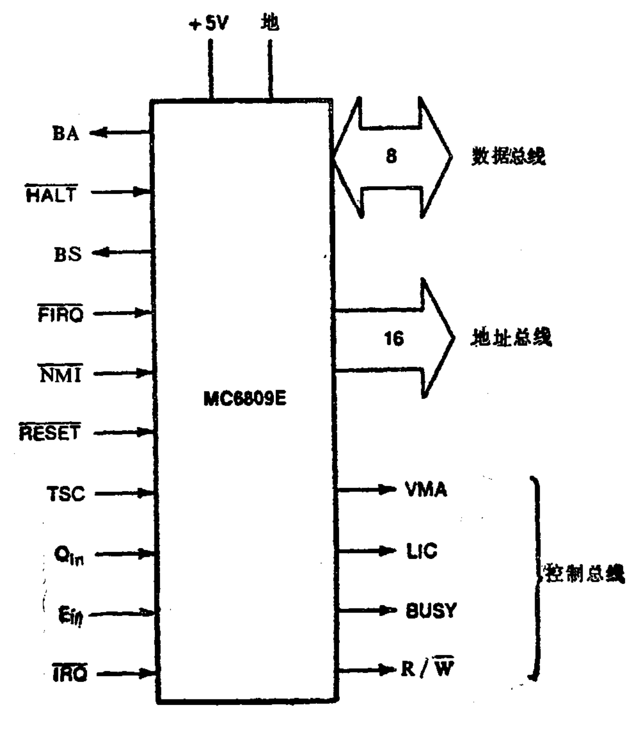


图1.8 6809E总线和控制信号

（4）设有存储器准备好的启动线MRDY，可延长数据存取时间，因此适宜于与慢速存储器配合工作。

（5）除规定的七种中断向量入口地址外（见第二章表2.9），还保留有一个＄FFF0＄FFF1地址单元可以作为外部设备的中断程序向量入口地址。

（6）设有同步响应回答输出信号（见第二章表2.8），允许和外部来的同步信号互相同步地进行工作。

（7）设有中断响应输出信号，表示处理器接受了中断处理（见第二章表2.8）。

（8）6809寄存器的结构比6800有了增加，如增加了第二个16位变址寄存器Y、第二个16位用户堆栈指示器U、一个8位直接页面寄存器DPR（使用户可以按零页面寻址方式向任何一个页面存取数据）。

（9）两个8位累加器可连接成为一个16位累加器D，因此可以执行16位算术逻辑操作和传送、交换操作。

（10）能及早确定低速存储器可使用的地址；动态存储器可使用的写入数据也能及早确定。

（11）在RESET复位输入端设有施密特触发器，故清零启动工作不需设置RC电路，只要加上电源即可在一个总线时间周期之内归零。归零后，到置位堆栈指示器为止，NMI自动被禁止工作。

（12）取消了6800中设置的VMA（存储器地址有效）线，选片时可不必再加相应的控制电路。

（13）加上存储器管理单元MMU时，可以具有2M字节的存储器寻址空间。

在E型的6809中，取消了晶体振荡电路输入端，设有BUSY和LIC（Last Instruction Cycle）输出端，因此可以知道其它处理器的状态。BUSY信号可以知道该处理器在使用存储器，LIC信号可以知道指令结束时所处的周期，故可以作为同步信号使用。在关系到安全第一采用双机运行保证高可靠的系统中或者在I/O和主系统等分散处理系统以及需要进行快速实时处理等系统中均可采用E型6809做成多处理器系统。①

在6809E中仍有VMA和TSC（三态控制线）。采用6809E的多处理器系统中，几个处理器为了利用同一硬件资源，需在一个周期之前知道下个总线状态，因此还要输出AVMA信号（Advanced Valid Memory Address）。

总之，莫托罗拉公司对6809在硬件和软件方面都作了许多重要改进。而为6800和6801编写的程序，只要稍做改变并符合下列各项条件，仍可在6809上运行。

（1）程序必须再重新汇编一次，因为不是6800/01所有操作码都和6809相同；

（2）只能使用标准堆栈，换句话说，除简单中断和子程序调用外没有其它堆栈；

（3）不用地址FFF0～FFF7；

（4）软件定时循环的要求不严格，因为指令周期的时间不同；

（5）条件码寄存器的高二位不能为1。

最后给出表1.8，列出6800、6801、6802、6809的基本性能比较，以便加深对6809性能的理解。

① Multiprocessing with Motorola's MC6809E
July 1981 BYTE

表 1.8　MC 6800/01/02/09 性能比较

6800	VMA	延迟时钟 慢Mem	01,02	堆栈指示器操作	中断进栈	CCR	中断向量	指令系统	16位比较: X寄存器	TST时条件码位	ASR、LSR、ROR时 条件码位	ASLA ASLB LSRA LSRB
6802	同 6800	存储器准备好输入 慢Mem	晶体输入 ÷ 4	同6800	同6800	同6800	同6800	同6800	同6800	同6800	同6800	同6800
6801	地址总线和R/W全部为 1	延迟时钟 慢Mem	-6801- 晶体输入÷ 4 -6801E- 启动输入	同6800	同6800	同6800	FFF6-7是输入捕捉 FFF4-5是输出比较 FFF2-3是时间溢出 FFF0-1是UART	机器码向上兼容，机器周期数少	C.V.N.Z 都受 X寄存器影响	同6800	同6800	同6800
6809	地址总线和R/W全部为 1 -6809E- 同6800	-6809- 存储器准备好输入 -6809E- 延迟时钟 慢Mem	-6809- 晶体输入 EOUT,QOUT -6809E- EIN,QIN	PUSH-减1并存储,PU-LL装入并加 1	-5个以上字节 -A和B都交换 -FIRQ只存PC和CCR	-第 6 位F屏蔽 -第 7 位E(全部)	FFF6-7是FIRQ FFF4-5是SW12 FFF2-3是SW11 FFF0-1保留	汇编语言向上兼容，机器周期数少	C.V.N.Z 都受 X.Y.U.S寄存器影响	C位不受影响	V位不受影响	同6800

第二章　6809硬件结构

2.1　6809电参数的最大额定值和电气指标

6809是8位微处理器，采用高密度N-沟、硅栅工艺，称为HMOS器件。有三种不同的封装形式，带有L、P、S标记分别表示为陶瓷封装、塑料封装和浸瓷封装。每种封装都有三种频率范围，分别为1.0MHz、1.5MHz、2.0MHz并表示为6809、68A09、68B09。其外部引线如图2.1所示。输入输出线设置情况如表2.1所示。

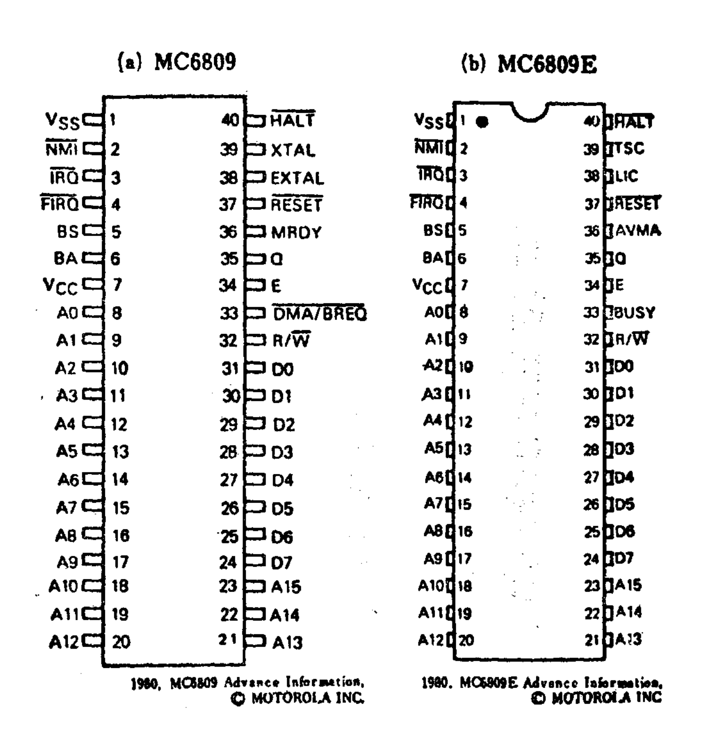


图2.1　6809/6809E外部引线配置图

1.　电参数的最大额定值

6809器件本身具有防止高静电电压或电场的保护电路，但在使用时仍应预防发生这种情况；而且应用过程中还应避免把超过最高额定值的电压加到这种高阻抗的电路上。在系统中没有用上的输入线应加上适当的逻辑电平（即V_{ss}或V_{cc}），这样可以提高工作的可靠性。

6809的电参数的最大额定值如表2.2所示。

2.　功耗和散热处理

6809/09E的最大功耗为1W。塑封外壳的热阻Q_{1A}最大值是100°C/W。所谓100°C/W的热阻的含意是：为了使功耗为1W的器件散热，从MOS晶体管（芯片）到外壳周围的温度

表 2.1 输入输出信号线

外 部 引 线	6809	6809 E
A0～A15	○	○
D0～D7	○	○
$\overline{HALT}$	○	○
$\overline{RESET}$	○	○
$\overline{NMI}$	○	○
$\overline{FIRQ}$	○	○
$\overline{IRQ}$	○	○
R/$\overline{W}$	○	○
Q	出	入
E	出	入
BA	○	○
BS	○	○
$\overline{DMA/BREQ}$	○	×
MRDY	○	×
XTAL	○	×
EXTAL	○	×
BUSY	×	○
AVMA	×	○
LIS	×	○
TSC	×	○

○ 表有信号线

× 表无信号线

表 2.2 最大额定值

额 定 参 数	符 号	数 值	单 位
电源电压	V_{cc}	−0.3～+7.0	V
输入电压	V_{in}	−0.3～+7.0	V
工作温度范围	T_A	T_L～T_H	
MC6809/09E, MC68A09/A09E, MC68B09/B09E		0～+70	℃
MC6809/09EC, MC68A09/A09EC, MC68B09/B09EC		−40～+85	
贮藏温度范围	T_{stg}	−55～+150	℃

差最大为100°C。当周围环境温度（T_A）为50°C时，则芯片表面在最坏状态下T_J＝150°C。当系统放在机箱中没有致冷风扇时，即使在有空调的房间内，芯片环境温度也很容易达到50°C。陶瓷封装的外壳因其Q_{JA}＝50°C/W，所以芯片温度在塑封外壳中若能达150°C，则陶封外壳的器件即可以放在100°C的环境中。

由于超大规模集成电路和大规模集成电路的可靠性受芯片表面温度的影响很大，所以使用环境应该限制在100°C以下，用致冷风扇冷却，或者使用散热片冷却。如果忽视了对于散热的设计，就会降低整个系统的可靠性，这是造成误动作的原因，应该绝对避免这种情况。

3. 电气性能

了解和掌握6809的电气性能是系统设计中的重要工作。其电气性能如表2.3所示①。在微

$V_{CC}=5.0V\pm5\%$

$V_{SS}=0$

表 2.3 6809电气性能 $T_A=0\sim70℃$

特　　性		符　号	最　小	典型值	最　大	单　位
输入高电平	逻辑，EXTAL $\overline{RESET}$	V_{IH} V_{IHR}	$V_{SS}+2.0$ $V_{SS}+4.0$	— —	V_{CC} V_{CC}	V
输入低电平	逻辑，EXTAL,$\overline{RESET}$	V_{IL}	$V_{SS}-0.3$	—	$V_{SS}+0.8$	V
输入漏电流 ($V_{in}=0\sim5.25V,V_{cc}=max$)逻辑		I_{in}	—	—	2.5	μA
直流输出高电平 ($I_L=-205\mu A,V_{CC}=min$)Do～D7 ($I_L=-145\mu A,V_{CC}=min$) Ao～A15,R/$\overline{W}$,Q,E ($I_L=-100\mu A,V_{CC}=min$) BA,BS		V_{OH}	 $V_{SS}+2.4$ $V_{SS}+2.4$ $V_{SS}+2.4$	 — — —	 — — —	V
直流输出低电平 ($I_L=2.0mA,V_{CC}=min$)		V_{OL}	—	—	$V_{SS}+0.5$	V
内部功耗（在稳态运行中，$T_A=0℃$时测试）		P_{INT}	—	—	1.0	W
电容量＊ ($V_{in}=0,T_A=25℃,f=1.0MHz$)	Do～D7,$\overline{RESET}$ 逻辑输入端，EXTAL, XTAL	C_{in}	— —	10 10	15 15	pF
	Ao～A15,R/$\overline{W}$,BA,BS	C_{out}	—	—	15	pF
工作频率	MC6809 MC68A09 MC68B09	f_{XTAL}	0.4 0.4 0.4	— — —	4 6 8	MHz
三态（断开状态）输入电流 ($V_{in}=0.4\sim2.4V,V_{CC}=max$)	Do～D7 Ao～A15,R/$\overline{W}$	I_{TSI}	— —	2.0 —	10 100	μA

＊电容量是定期测试的结果

型计算机的硬件系统设计中，需要着重考虑的问题之一就是微处理器与集成电路器件和外围器件相互连接的问题。系统设计人员应该分析了解控制总线、地址总线和数据总线的接口连接性能。需要知道6809中各条总线的性能和作用。当接到总线上的器件数目超过规定数时，微机的系统设计就会复杂一些。当系统按最小组成配置时，系统设计就简单易行。但在使用慢速器件或大量集成电路时，就需要改变设计。所以在不了解MPU信号的电气性能时是不能做好系统接口设计的。

（1）地址总线特性

如图2.2所示，地址总线在Q脉冲的上升沿处有效。通常可用该上升沿把地址内容锁存到

---

① 6809E的电气性能见附录

外围器件中去，如ROM、RAM。地址总线有一定的负载要求，必须遵守不能违犯。每条地址线可以驱动一个肖特基TTL负载和90pF的电容。因为大多数68系列的器件的输入电容量接近10pF，所以最小组成的6809系统在规定的时钟频率之下保证可驱动8个68系列器件和一个肖特基TTL器件（不需要缓冲驱动器）。在低速时钟频率时可以驱动更多一些器件，但设计时要仔细地测试和分析。当需要使地址总线增加驱动能力时可以加入缓冲驱动器，可采用8T97、SN74240、SN14244等器件，如图2.3所示。

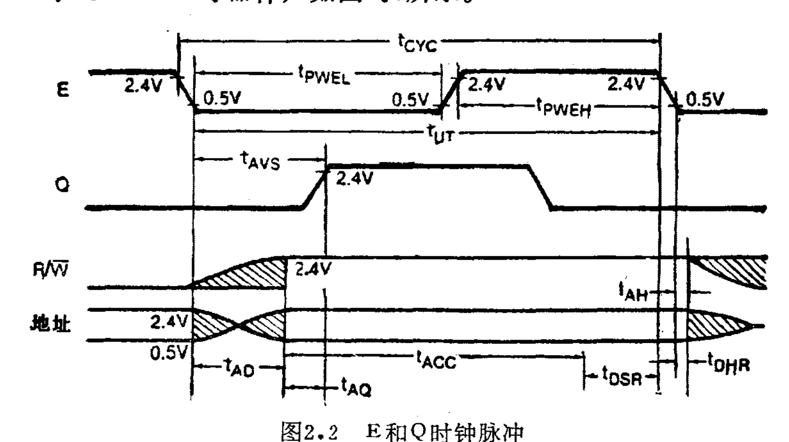


图2.2 E和Q时钟脉冲

注：所有输入、输出波形测量规定，逻辑高电平为2.0V，逻辑低电平为0.8V

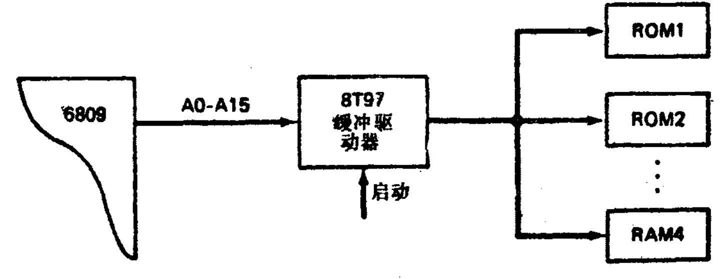


图2.3 地址总线缓冲驱动器

（2）数据总线特性

数据总线的负载性能和地址总线类似，但不完全相同。MPU数据线的最大负载是一个肖特基TTL负载加上130pF。因为68系列器件输入端接近10pF，故可驱动12个68系列的器件。

要具体说明的几个问题如下：

其一，在满足一种特定负载配置要求时，首先根据6809的技术条件确定最坏情况下的电流要求，进行逻辑“1”和“0”电平时电流I_{OH}、I_{OL}的分析，确定直流（静态）负载；其次，在所要求的工作频率之下，得到地址总线和数据总线的最大允许电容负载，即交流（动态）负载；最后，把所有外部器件要求的负载电流和电容加在一起，验证是否在6809技术条件之内。

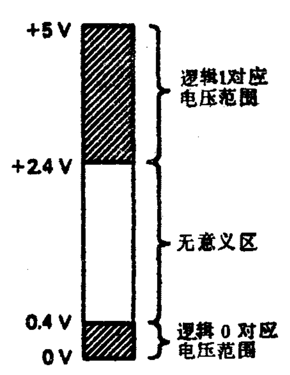


图2.4 TTL器件输出电压范围

另外，不论何种工艺的器件其信号电压都有一定的有效工作区，而且也都有一个不确定的区域。TTL器件的输出范围如图2.4所示。同时，根据6809的电气性能可以知道其电源电压V_{CC}

为5V±5%。6809输出口在最坏情况下等效电路如图2.5所示。

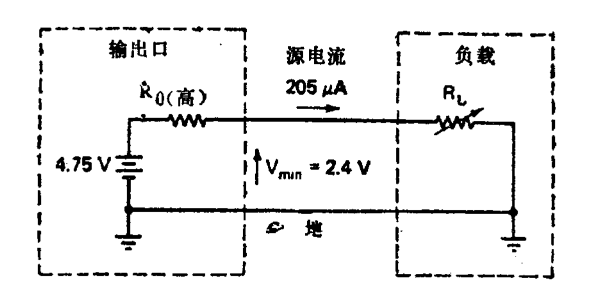
其二，数据总线的负载不是在最小情况下，检验负载的问题。这时假设系统中需要6片ROM、2片RAM和4片PIA（68系列并行接口）器件，问6809工作在1MHz时不用总线驱动器可否？回答是不行的。因为6809在最小配置下只可驱动8片68系列器件和一个TTL负载。

其三，数据总线的负载检验问题。问6809输出线可驱动多少74LS系列的TTL输入端？对此应做如下分析：根据6809电气性能可知D0～D7的输出端在不能再区别逻辑1之前，处在逻辑"1"电压V_{OH}时可以提供的最大电流是：

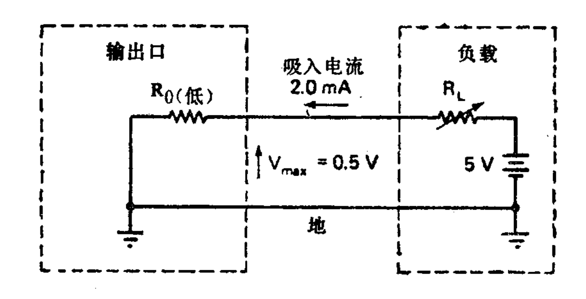
$$I\ (\text{load}) = 205\ \mu A$$

如果6809的每条输出线的输出电流超过205 μA，那么输出电压就可能低于2.4V。

SN74系列器件的输入负载电流和输出驱动电流的数值见表2.4和表2.5。根据表



(a) 输出高电平



(b) 输出低电平

图2.5 6809输出口在最坏负载的情况

表2.4 74系列输入负载电流

逻辑系列		74LS××	74××	74S××
1	输入端为2.4V时最大电流	20μA	40μA	50μA
0	输入端为0.4V时最大电流	0.36mA	1.6mA	2.0mA

表2.5 74系列输出驱动电流

逻辑系列		74LS××	74××	74S××
1	输出端为2.4V时最大施出电流	400μA	400μA	1000μA
0	输出端为0.4V时最大吸入电流	9 mA	16mA	20mA

2.4所示，每个74LS××可以接收20μA电流，即

$$20\mu A \times 10 = 200\mu A < 205\mu A$$

或者说6809输出线为逻辑1电平时可使10个74LS××输入端工作。在逻辑0电平时，74LS××将给6809回授0.36mA。因此

$$0.36mA \times 5 = 1.90mA < 2.0mA$$（6809在0电平时的最大吸入电流）

所以只能有5个74LS××器件工作，不是10个。

(3) 控制信号线特性

所有微处理器为使外部器件工作都设有一系列控制信号。属于工作的控制信号有总清 $\overline{RESET}$、暂停$\overline{HALT}$、请求总线 $\overline{DMA/BREQ}$；属于中断信号有非屏蔽中断 $\overline{NMI}$、快速中断请求$\overline{FIRQ}$、和中断请求$\overline{IRQ}$；属于访问存储器的信号有读写 R/$\overline{W}$、存储器准备好MRDY。此外还有状态信号，可用总线 BA 和总线状态 BS，以及时钟信号E、Q。以上这些信号线详细内容将在2.3节中叙述。

4. 总线定时性能

6809的总线上信号的时间特性如表2.6所示。① 总线定时的时间关系如图2.6所示。

表2.6 6809 总线定时特性 (注1,2)

识别号	特　　性	符　号	MC6809 Min	MC6809 Max	MC68A09 Min	MC68A09 Max	MC68B09 Min	MC68B09 Max	单位
①	周期时间（注5）	t_{cyc}	1.0	10	0.667	10	0.5	10	μs
②	脉冲宽度，E低	PW_{EL}	430	5000	280	5000	210	5000	ns
③	脉冲宽度，E高	PW_{EH}	450	15500	280	15700	220	15700	ns
④	时钟上升和下降时间	t_r, t_f	—	25	—	25	—	20	ns
⑤	脉冲宽度，Q高	PW_{QH}	430	5000	280	5000	210	5000	ns
⑥	脉冲宽度，Q低	PW_{QL}	450	15500	280	15700	220	15700	ns
⑦	延迟时间，E到Q上升沿	t_{AVS}	200	250	130	165	80	125	ns
⑨	地址保持时间（注4）*	t_{AH}	20	—	20	—	20	—	ns
⑩	BA、BS、R/$\overline{W}$和地址有效时间到Q上升沿	t_{AQ}	50	—	25	—	15	—	ns
⑰	读数据建立时间	t_{DSR}	80	—	60	—	40	—	ns
⑱	读数据保持时间*	t_{DHR}	10	—	10	—	10	—	ns
⑳	从Q算的数据延迟时间	t_{DDQ}	—	200	—	140	—	110	ns
㉑	写数据保持时间*	t_{DHW}	30	—	30	—	30	—	ns
㉙	有效存取时间（注3）	t_{ACC}	695	—	440	—	330	—	ns
	处理器控制建立时间（MRDY、中断、$\overline{DMA/BREQ}$ $\overline{HALT}$、$\overline{RESET}$	t_{PCS}	200	—	140	—	110	—	ns
	晶体振荡器起始时间	t_{RC}	—	100	—	100	—	100	ms
	处理器控制上升和下降时间	t_{pcr}, t_{pcf}	—	100	—	100	—	100	ns

*地址、数据的保持时间是周期性测得的，不是100%测试

注：

1. 除非另有说明，所有电平都是$V_L \leqslant 0.4V, V_H \geqslant 2.4V$
2. 除非另有说明，所有测试点都是0.8V和2.0V
3. 可用的存取时间由1－4－7 max＋10－17计算
4. BA，BS保持时间（⑨）没有规定
5. 在MRDY或$\overline{DMA/BREQ}$期间最大t_{CYC}是16μs

① 6809E总线定时特性见附录

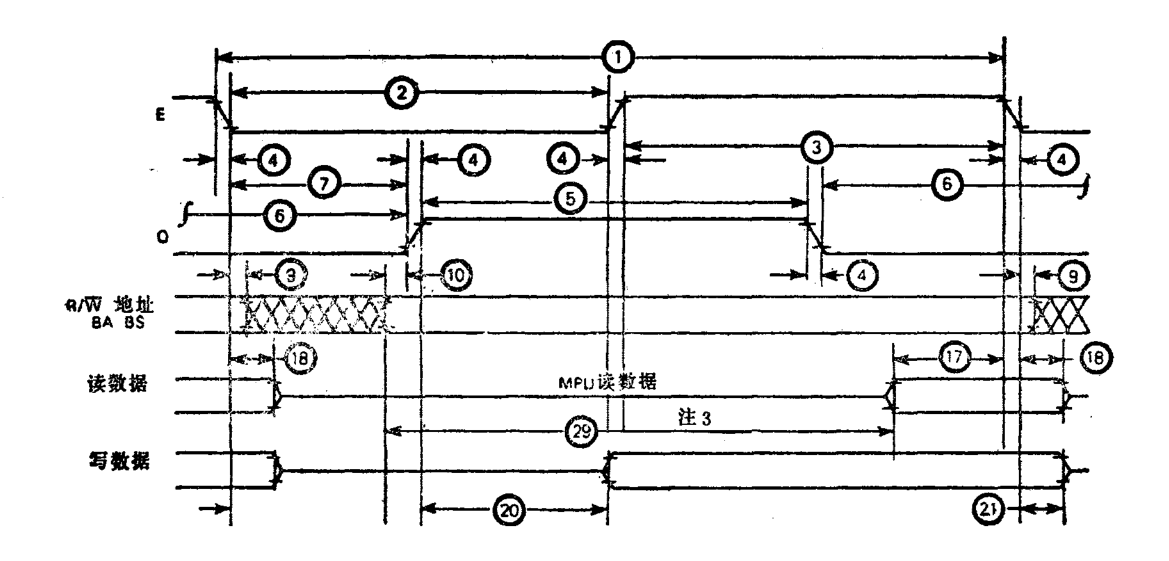


图2.6　总线定时图（图中识别号和注见表2.6）

微处理器系统速度减慢的主要原因是电容的作用，从而影响了定时要求。因为许多器件都要接到地址和数据总线上，所以使总线的电容增加。由于MPU需要驱动较重的负载，因此减慢了动态响应的时间，造成了延迟时间（信号达到其终值50%时的间隔时间）。6809输出信号线上受轻载或重载的影响状况如图2.7所示。在50%的点上要能达到电压的门限值，接收的器件在有杂音的情况下可以区别 0 和 1 之值。

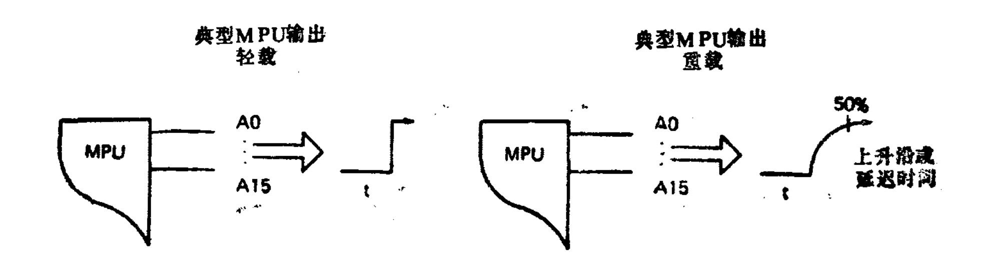


图2.7　电容性负载的影响

（1）MPU定时与各器件匹配的方法

为使MPU定时与各器件相匹配应按以下步骤进行：

- 为了估计电容的影响和准确地决定时钟周期，需把各器件的总电容负载相加；
- 计算其它有关电路通路的延迟时间；
- 为了确定地址、数据和读写信号能允许的最坏情况的延迟时间，应参考如图2.8所示的MPU的交流负载曲线关系图；
- 如果需要减慢MPU，根据软件安排合理地使用MRDY或$\overline{\text{DMA/BREQ}}$；
- 使MPU的时钟速率降低到实测要求的水平，或者选择更快速的6809，如68A09、或68B09。

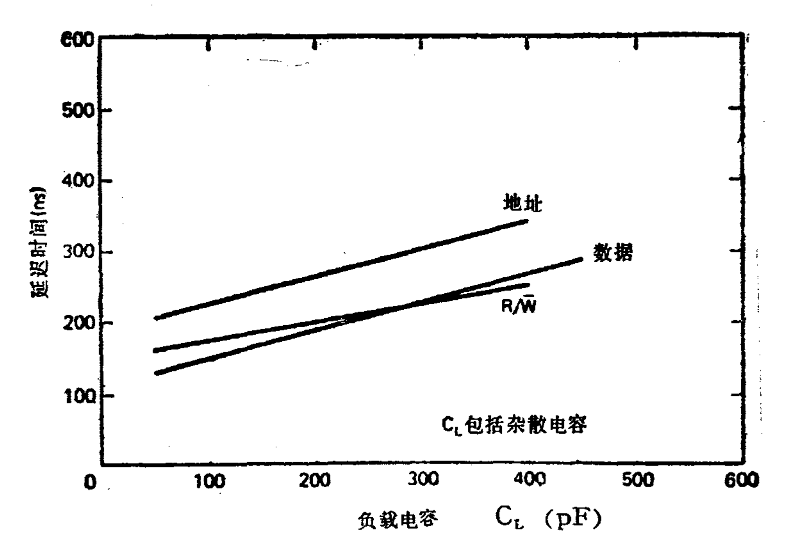


图2.8 MPU的交流负载近似曲线

图2.8所给的负载和延迟时间关系曲线是典型的情况，它适合于许多微处理器使用。注意使用时所说的准确的工作条件（I_{OL}、I_{OH}，即输出低电平的电流、输出高电平的电流）。在每条曲线下的全部区域才是有效工作区。所以，用该曲线中的数据即可决定数据线和地址线上负载电容的影响。当增大负载时，即增加负载电容，因此，数据和地址信号的时间也就按比例地延长。

（2）三态总线的考虑

在许多实际应用的系统中需要把大量的各种器件接成总线式结构。大多数微处理器系统中都考虑设有三种总线：即地址总线、数据总线和控制总线。如果采用数据缓冲器来接受这些时间延迟，那么就可以减少负载的问题。但是，当前并不要求使用数据或地址缓冲器。

这样就产生一个很重要的问题：怎样才能使许多器件共用同一条总线？为解决此问题，微处理器逻辑设计人员设计出一种称为三态的逻辑电路，这种器件的输出不是处在工作状态（1态或0态）就是处在高阻抗（开路）状态。因此，把许多器件连到一条总线上时，就只有一个器件在工作，而其余的器件都保持在高阻抗状态。三态逻辑的目的是使器件与微处理器系统隔离，以便建立起一条共用的总线。

因此必须提供一种控制信号使器件本身可处于高阻抗状态。在6809设计中，数据总线可以采用双向总线驱动器作为三态器件，如8T26、8T28等。地址总线无需使用双向总线驱动器（只有在DMA时才用）。

6809的产品说明书中有三态的说明。从技术说明书中可以看到，6809的三态(开路状态)的输入电流 I_{tsi} 是满足系统设计要求的。其中最大的漏入6809数据线电流为10μA；而地址线的漏电流为100μA（仅在这些线为高阻抗状态时）。

怎样才能保证三态情况下不会超过负载的要求？对这种工作状态和开路状态的分析相当简单。每个逻辑器件在高阻抗状态时都有一个可以允许的输出电流，类似于以前的分析，现说明如下：

工作状态分析

• 当所有器件都处在开路状态时，求出所有输入电流之和；

• 说明在使用数据总线驱动器时，或者直接连到6809数据线上时，该电流都不超过最大允许数值。

6809的数据线可以对工作的器件或开路状态的器件给出205μA的电流；6809的地址线可以给出145μA电流。

高阻抗状态分析

重复上述步骤1和2，但数据线用10μA、地址线用100μA。三态设计中主要的考虑决定于地址总线和数据总线的负载。然而当所设计的微处理器系统小于系统的最小配置时，可以不必进行三态分析。这时系统设计的分析就更为简单。

（3）测试总线定时的负载

6809芯片测试线路如图2.9所示。

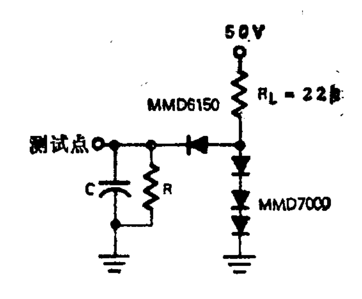


图2.9 总线定时测试负载
C= 30pF 对BA，BS
130pF对D0～D7、E、Q
90pF 对A0～A15、R/$\overline{W}$
R= 11.7kΩ对D0～D7
16.5kΩ对A0～A15、E、Q、R/$\overline{W}$
24kΩ对BA，BS

2.2 6809的内部结构

每个微处理器系统都有两种结构，即硬件体系结构和软件体系结构。微处理器的内部结构（即框图）通常是指微处理器本身的硬件结构。其中应包括寄存器和运算逻辑单元，以及所采用的总线。

微处理器框图可以使我们了解硬件体系结构的特性，框图中给出的内部寄存器，虽然有的不能直接给用户使用，但是可以帮助我们了解MPU的功能和某些指令的作用。

微处理器的软件的体系结构是用程序设计模型来说明的。类似于框图的程序设计模型可以告诉程序设计人员直接能够使用的各种寄存器和处理器的其他部件。程序设计模型中通常很少表示额外的资源。这种剥离型软件体系结构就象硬件体系结构一样可以帮助我们迅速掌握微处理器程序设计的特点。在本部分中将要说明6809的内部结构框图和程序设计模型。

1. 6809的内部结构框图

6809是在硬件特性方面较早期机型有很大改进的一种第二代微处理器。采用了简单的时钟、复位总清和其它控制信号。其内部结构框图如图2.10所示。

根据该内部框图可有助于迅速确定可能的指令并掌握内部体系结构。例如有下述几点可以确定：

• 从堆栈指示器S到D0～D7之间有连接线这就意味着在指令系统中存在着可以这样使用的某些指令；

• 直接页面寄存器DPR被连到地址线上，这就说明DPR可以作为一个源地址使用；

• 除A和B累加器之外没有设置通用寄存器。累加器A和B可以连接成累加器D；

• 易于识别控制信号的功能。

2. 6809的程序设计模型

任何一个微处理器都需具有的两个重要特性就是硬件能力（电气性能）和软件的能力。根据程序模型就可以了解存在于特定微处理器中的软件能力。6809的程序设计模型如图2.11

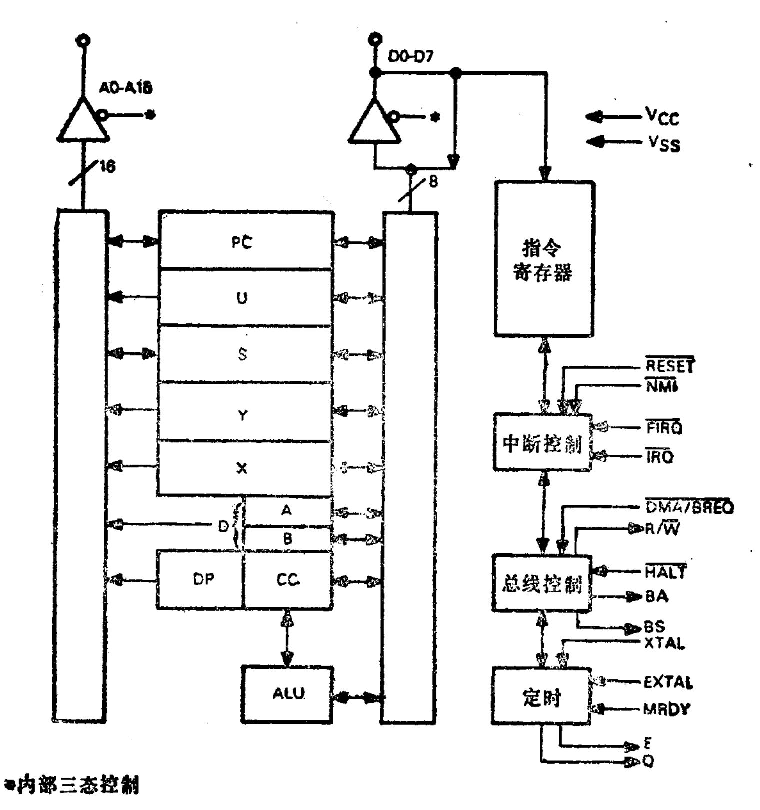


图2.10　6809内部结构框图

所示。6809比6800增加了三个寄存器，它们是直接页面寄存器、用户堆栈指示器和第二个变址寄存器。

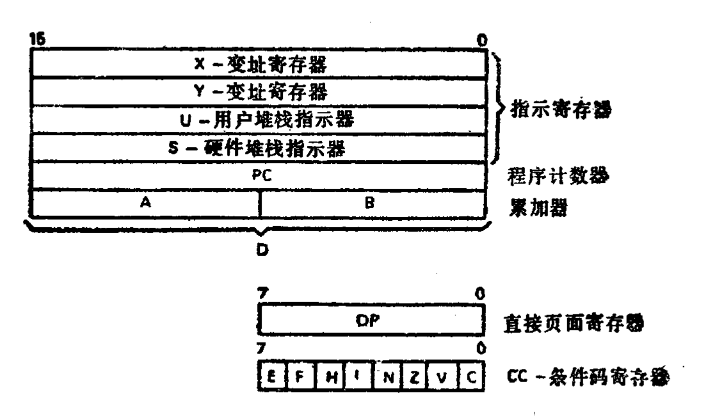


图2.11　6809的程序设计模型

6809的程序设计模型在硬件上有许多微妙的改进，它们在新产品和系统的设计中起了重要的作用。其中主要的有：

- 两个16位变址寄存器（X、Y）；
- 两个堆栈指示器（U、S），U堆栈指示直接给用户使用；
- 有一个双字长累加器D，实际上是累加器A和B联合在一起处理；
- 有一个直接页面寄存器DPR（在每次RESET时都被清零）；

• 有两个CCR位：用F位屏蔽FIRQ中断，用E位规定中断时寄存器进栈的条件；
• 有多种寻址方式：间接寻址、前/后缀变址、程序计数器相对寻址PCR。

以上这些新资源的应用见表2.7

表 2.7 新资源的应用总结

应　　　用	资　　　源
位置独立程序	PCR、长分支转移
再入程序；	DP、U
模块化程序设计	DP、U、X、Y
堆栈机器	DP、E、$\overline{\text{FIRQ}}$、U、S
多微处理技术	DP、U、S、$\overline{\text{FIRQ}}$

（1）累加器（A、B、D）

A、B寄存器作为通用累加器，用来进行算术运算和数据处理。

某些指令可把A、B寄存器连接起来形成一个16位累加器，称为D寄存器，其中A寄存器为高位字节、B寄存器为低位字节；

（2）直接页面寄存器（DPR）

6809的直接页面寄存器为增强直接寻址方式的能力而设。在直接寻址指令执行期间，该寄存器的内容出现在高位地址的输出线上（A8～A15）。这样，直接方式在程序控制下可用在存储器中的任何地址上。在处理器总清复位期间，该寄存器所有各位都被清零，所以保证同6800兼容。

另外要说明的是该DP寄存器内容不能轻易地被改变，这是故意这样设计的，因而没有直接立即装入DP的指令。为了改变DP的内容，必须先把内容装入另一个8位寄存器并执行一次交换或传送操作后才能进入DP。

（3）变址寄存器（X、Y）

变址寄存器用作变址寻址方式，该寄存器中的16位地址都参加有效地址的计算。该地址可以用来直接指出数据，或者由所选的常数或寄存器偏移值来修改。在某些变址方式中，变址寄存器的内容可以被增加或者减少，以便指到表格式数据的下一项。X、Y、U、S所有这四个指示寄存器都可作为变址寄存器使用。

（4）堆栈指示器（U、S）

在子程序调用和中断过程中，处理器可以自动地使用硬件堆栈指示器（S）。6809的堆栈指示器是指出堆栈的顶部单元，与6800的堆栈指示器不同，后者是指出堆栈的下一个自由的单元。用户堆栈指示器（U）只能由程序人员控制使用，所以很容易使自变量出入子程序。和X、Y寄存器一样这两个堆栈指示器都有相同的变址寻址能力，但它们还支持压入和弹出指令（即进栈或出栈指令）。因此6809可以很有效地作为堆栈处理器，极大地增强了支持高级语言和模块化程序设计的能力。

下面说明有关堆栈和堆栈操作的两个问题。

堆栈问题　在所有应用中通常都要求使用很少几条指令即可迅速地和自动地把信息存

储起来。那么使用带有堆栈工作能力的一条指令即可满足要求。中断本身特别需要这种能力。6809堆栈中寄存器保留的顺序如图2.12所示。进栈操作是把MPU中的寄各存器按递减存储器单元的顺序(向单元零的方向)存入存储器，出栈操作方向相反。

6809进出堆栈的操作还有一个节约时间的好性能。当不需要把所有MPU寄存器保留在堆栈中时，可以利用堆栈指令的第二个字节实现，即用后缀字节规定所要保留的寄存器。

堆栈操作问题 模块化的程序是非常好的程序形式，之所以设计成这样的程序是因为它易于调试和维护。多数情况下是利用子程序作程序模块。而这种结构在6809中极易于实现。堆栈过程的一致性是非常重要的，因为对于子程序来说参数的传递和局部的存储都是不可缺少的。6809中利用堆栈的情况如图2.13所示。应该注意在调用子程序之前,调用程序要建立堆栈中的参数单元。按照调用子程序的要求，子程序应该为局部变量建立堆栈空间（仅在需要时）。

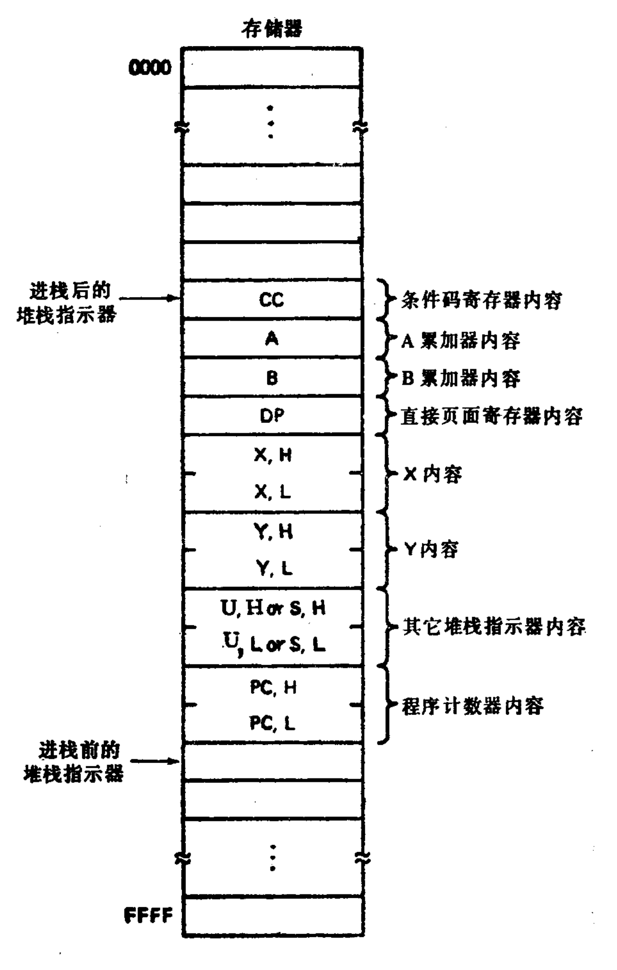


图2.12 6809进栈的顺序

各个堆栈的一致性的处理是非常重要的。为了做到这一点有两种不同的但过程是可靠的方法。第一，在调用的程序中，保留原来堆栈标记、U为子程序产生参数空间(LEAS—N，U），显然S堆栈指示器在参数空间之上。第二，在进入子程序时为了立即保留原来的MPU寄存器内容，这时可由子程序本身来改变（PSHS B，A，CCR）。也就是由子程序来产生局部存储所需要的空间。在图2.13中可以看到需要保留B，A和CCR的内容，这是因为子程序将要改变这些寄存器的内容，也就是说要设置4个局部存储单元（LEAS—4,5）。

（5）程序计数器（PC）

处理器使用程序计数器是为了指出由处理器将要执行的下一条指令的地址。程序计数器本身可以进行相对寻址，在某些情况下类似使用变址寄存器一样。

程序计数器是一个16位的二进制计数器。程序计数器本身还参加分支转移相对寻址的转移目的地址的计算和程序计数器相对存储器寻址方式的地址的计算。

由于6809的程序计数器可以实现相对寻址，因此可以编写位置独立程序，减少软件开销。同时数据本身也可以放在6809存储器中的任何地址区上，实现完全的位置独立程序。

（6）条件码寄存器（CCR）

条件码寄存器表示在任何所给的时间之内处理器的状态。条件码寄存器各位的含意如图2.14所示。

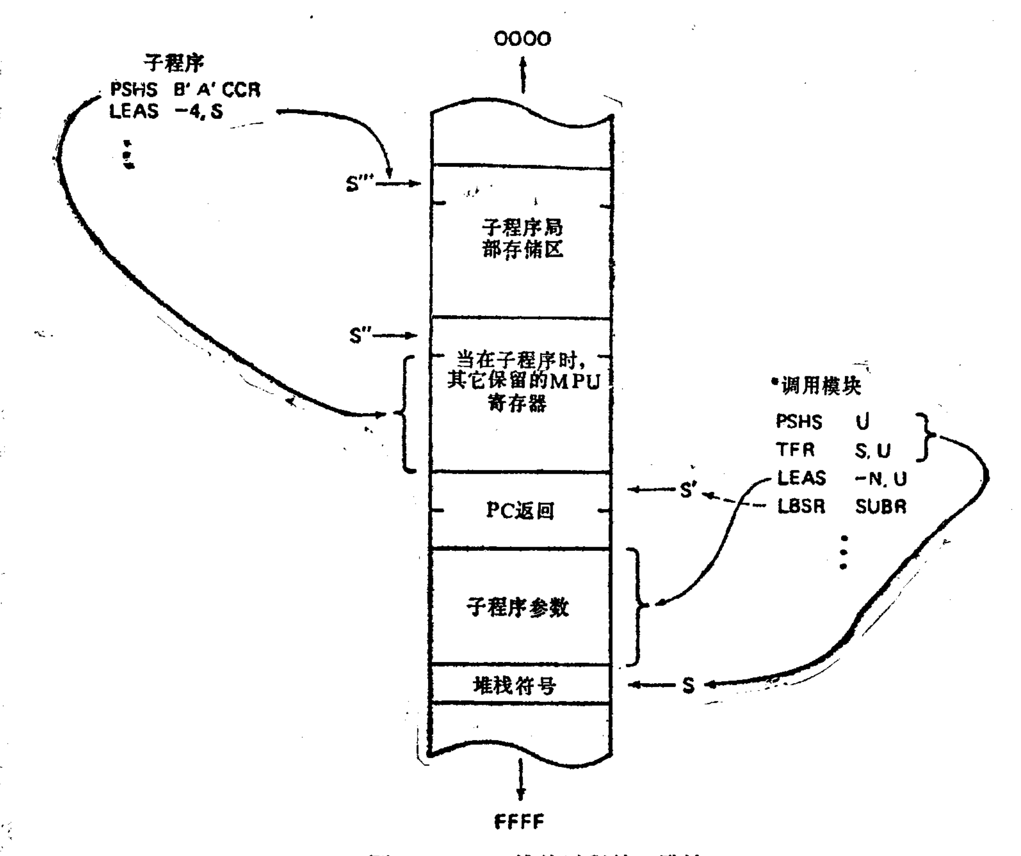


图2.13 6809堆栈过程的一致性

条件码寄存器各位的含意说明如下：

E F H I N Z V C
进位
溢出
零
负
IRQ屏蔽
半进位
FIRQ屏蔽
全部标志

图2.14 条件码寄存器的格式

0位（C） 0位是进位标志位，通常是表示从二进制算术逻辑部件ALU来的进位。也用C来表示执行减法的指令（CMP、NEG、SUB、SBC）产生的借位，并表示二进制ALU部件产生的补码位。

1位（V） 1位表示溢出位，当带符号的2的补码运算溢出时，该位置1。当ALU中最高位MSB产生的进位与MSB-1位产生的进位不一致时，要检查该溢出位的状态。

2位（Z） 2位表示零位，当前面的运算结果确定为零时，该位被置1。

3位（N） 3位表示负数位，前面运算结果的最高位MSB的数值应包括在数值之内。所以一个负的2的补码结果出现时，N位将被置1。

4位（I） 4位表示$\overline{IRQ}$中断的屏蔽位，如果该位置1，处理器就不能识别$\overline{IRQ}$线上来的中断信号。$\overline{NMI}$、$\overline{FIRQ}$、$\overline{IRQ}$、$\overline{RESET}$和SWI都可使该位置1；SWI2、SWI3不影响I位。

5位（H） 5位表示半进位位，而且只是在用ADC或ADD指令作8位加法时，表示ALU部件中从第3位产生的进位。DAA指令用该位来完成BCD十进制加法的调整操作。在所有减法类的指令中该标志位的状态没有定义。

6位（F） 6位表示$\overline{FIRQ}$的屏蔽位，如果该位为1，处理器即不会识别从$\overline{FIRQ}$线

来的中断。$\overline{NMI}$、$\overline{FIRQ}$、SWI和$\overline{RESET}$各信号都将使F位置1；而$\overline{IRQ}$、SWI2和SWI3对F位没有影响。

7位（E） 7位是全部标志位，该位置1时，表示全部机器状态（所有寄存器）都要放入堆栈，而不是部分处理器状态（PC和CC）。在从中断返回过程中（RTI）要用到进栈的条件码寄存器中的E位，以决定出栈的范围。所以，被放进条件码寄存器中的当前的E位表示过去的动作。

2.3 6809的输入/输出信号线

在本节中将详细讨论6809和6809E的输入/输出信号。这两种MPU器件之间的差别即在于输入/输出信号有所不同，而不是指令系统不同。6809E不要连晶体，增加了两条状态线。用这两条线可以组成多处理器系统的应用。

为了方便起见，可以把这些信号按功能分为五大类。即是电源/时钟、数据/地址、总线状态、总线定时和控制五类。6809/09E所有这五类提供的能力都比6800系列有很大的增强。例如，用五条线译码就可以准确地确定正在取出的中断向量，并且可给I/O器件提供一个完整的中断响应回答。另外，6809的直接存储器存取（DMA）的能力由于增加了$\overline{DMA/BREQ}$控制线也比6800有很大的提高。因此我们将讨论在6809和6809E中使用的三种DMA方法，即暂停方式（Halt-Mode）、周期窃取方式（Cycle Stealing）、和总线多路转接方式（Bus Multiplexing）等DMA方式。

2.3.1 6809的引线

6809的引线配置情况如图2.15所示，在讨论引线的功能时，我们把引线分成五类功能，如前所述即电源/时钟、数据/地址、总线状态、总线定时和控制五类，现分述如下：

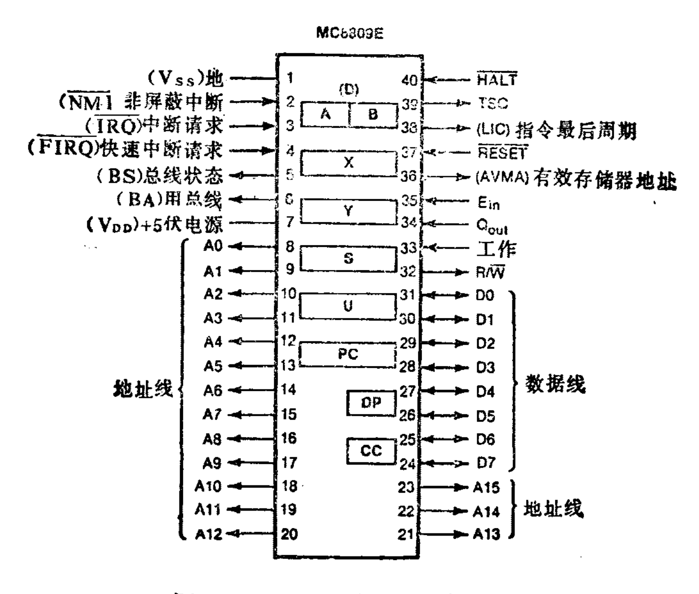


图2.15 6809的引线配置情况

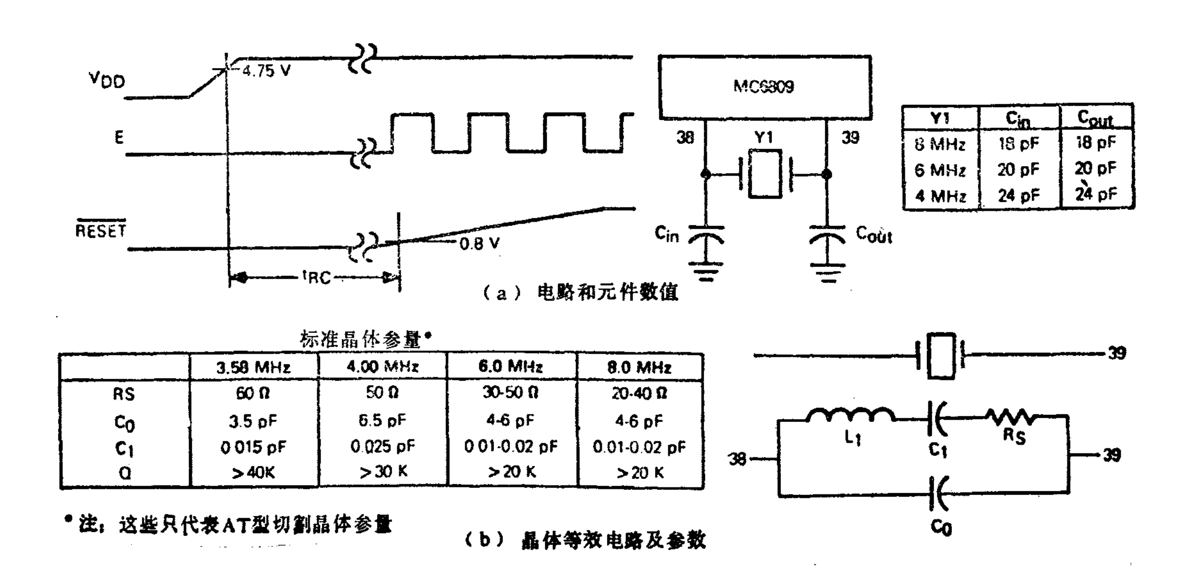
1. 电源/时钟线（V_{SS}、V_{DD}/XTAL、EXTAL）

和所有6800系列器件一样，6809需要用单一+5 Vdc±5%的电源电压供电，其最大电流消耗200mA，相应的最大功率损耗为1 W。第1引线V_{SS}接地，第7引线V_{DD}接+5 Vdc。

和6800不同，6809的芯片本身设有内部时钟振荡器/驱动器，但是必须在引线38和引线39（即EXTAL、XTAL）之间外加并联谐振晶体，以便对内部振荡器的频率提供晶体控制。另一方面，可以在EXTAL引线上加外部定时信号的TTL输入电平，而XTAL线接地。标准的6809最高工作频率为1 MHz，但内部电路需对外部晶体频率4倍分频。所以需使用4MHz晶体才能得到1 MHz内部工作频率。实际上也可以使用廉价的3.58MHz的电视彩色同步信号晶体，这时内部工作频率为0.895MHz，完全可满足许多应用。6809也有1.5MHz工频的68A09、和2.0MHz工频的68B09，用这两种工作频率时将分别需要6 MHz和8 MHz的晶体。

当前对速度问题，许多微处理器都自夸有很高的时钟频率，从理论上讲MPU的周期时间的提高会影响整个处理的速度。然而这只能在某种范围如此，所以只说时钟频率——即MPU周期时间是不够的，它只讲了控制处理器速度的一种因素。衡量整个速度的最好说法是处理器的吞吐量。该术语不但考虑了MPU的周期时间，而且还考虑了其它重要因素，如体系结构、指令系统的效率、寻址能力、总线定时等等问题。当同其它8位机系列比较时，考虑到这些因素在内，6809能处理的吞吐量是很高的。所以6809的内部时钟频率无需超过2 MHz就能达到与它竞争的处理器的性能，或者有超过它们的更好的性能。而在相同的时间内，避免了原有的高频带来的问题。在实际应用系统中如果需要超高处理速度，解决问题的办法是采用多处理器系统，而不是使处理器使用100MHz的时钟。总之，时钟频率最佳值的根据就是按应用要求设计。只要按要求时间处理器可完成工作，那么就作为处理器的最佳速度被确定下来，按这种方法考虑设计是非常重要的。

6809的晶体连接要求如图2.16所示，应注意的是两个电容器C_{in}和C_{out}要接到晶体引线



Y1	Cin	Cout
8 MHz	18 pF	18 pF
6 MHz	20 pF	20 pF
4 MHz	24 pF	24 pF

（a）电路和元件数值

标准晶体参量*

	3.58 MHz	4.00 MHz	6.0 MHz	8.0 MHz
RS	60 Ω	50 Ω	30-50 Ω	20-40 Ω
C0	3.5 pF	6.5 pF	4-6 pF	4-6 pF
C1	0.015 pF	0.025 pF	0.01-0.02 pF	0.01-0.02 pF
Q	>40K	>30 K	>20 K	>20 K

*注：这些只代表AT型切割晶体参量

（b）晶体等效电路及参数

图2.16 6809晶体连接方法

和地之间。这种连接可以保证器件有最佳性能。另外，为了减小畸变，按装晶体时应尽量靠近EXTAL和XTAL引线端，并应考虑按射频技术要求布置印制板排线，如图2.17所示。莫托罗拉公司建议不要使用LC网络来代替晶体工作，还说器件加电后只需20ms时间即可工作。

此外，还可以采用外部TTL或CMOS时钟信号驱动6809，使用时，该信号必须加到38引线上（EXTAL），而39引线（XTAL）接地。为建立内部工作频率，需使外时钟信号除以4。

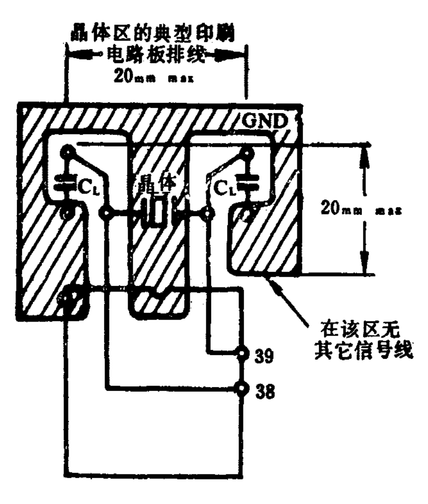


图2.17　6809晶体印制板布线图

2. 数据/地址线

（1）数据总线（D0～D7）

6809有8条双向数据总线作为系统的通信总线。每条数据线可驱动一个肖特基TTL负载或4个LS TTL负载和130pF电容。数据总线内部为三态驱动器。

（2）地址总线（A0～A15）

6809设有16条单向输出地址信息的地址总线。当处理器不需要数据传送总线时，地址总线输出地址将是$FFFF_{16}$、读/写信号$R/\overline{W}=1$、总线状态BS＝0，这种情况称为“假地址”或$\overline{VMA}$周期。地址线在Q时钟信号的上升沿开始有效，见图2.6。所有地址总线驱动器，在输出可用总线信号（BA）为高电平时将为高阻抗状态。每条地址线可驱动一个肖特基TTL负载或4个LS TTL负载和90pF电容。

在组成系统时，数据/地址线在额定总线速度下都可驱动一个标准的低功耗TTL负载和8个6800系列的器件。如果降低总线电容负载的影响，减少时钟速度，那么驱动6800系列的器件可以超过8个。

3. 总线状态（BA、BS）

6809设有两条使MPU与外围器件通信的状态线，即引线5为总线状态线（BS）和引线6为可用总线线（BA）。

（1）BA线

可用总线输出BA是内部控制信号，表示使MPU的MOS总线为高阻抗状态，即数据、地址和$R/\overline{W}$线都处于三态逻辑状态。如果BA＝0，表示处理器正在运行，而且数据线、地址线、读/写线的三态逻辑为工作状态；如果BA＝1，表示处理器处在暂停条件下，而且数据线、地址线、读/写线三态逻辑为断开状态，即处在高阻抗状态。所以处理器和外数据、地址总线完全断开，而使这些总线为外部器件使用。然而BA＝1不是只指数据、地址总线可多用一个MPU周期。当BA为低电平时，在MPU取得总线之前要经过一个死周期的时间。

（2）BS线

总线状态输出信号当其同BA一起译码即表示MPU的状态（在Q时钟的前沿），如表2.8所示。正常的运行状态显然无须进一步讨论，现只讨论其它几种状态。

中断确认（IACK）状态表示外部器件给6809的某一中断（$\overline{RESET}$、$\overline{NMI}$、$\overline{IRQ}$、$\overline{FIRQ}$、SWI1、SWI2或SWI3）已经被接受，这时表示经过两个周期把处于＄FFF2～＄FFFF各个存储器空间中相应于各个中断的向量地址的内容传送到程序计数器之中。显然

表 2.8　MPU 的状态

MPU 状态		MPU 状态的定义
BA	BS	
0	0	正常运行状态
0	1	中断或总清确认状态（取进向量地址过程）
1	0	同步确认状态（中断等待过程）
1	1	暂停/总线回答确认，6809E 只确认 $\overline{HALT}$

用IACK状态可以通知外部I/O器件说明中断已被接受。该状态可以在外部用逻辑译码来检查 BA＝0、BS＝1的状态。另外如果地址线 A1、A2和A3也同BA/BS一起译码，则译码逻辑可以完全表示中断已被接受。如图2.18所示即为采用 74154 译码的电路图，当向量地址被取出时，中断的向量地址即被放在地址总线上，因此表示中断已被接受的BA/BS逻辑和地址线 A1～A3将完全准确地表示中断向量正在被取出。因为SWI3向量地址为 FFF2:FFF3，所以74154的 2 线输出有效低电平；$\overline{IRQ}$向量地址为FFF8:FFF9，8 线输出有效低电平；如此等等。因为所有中断向量地址开始值均为偶数地址，所以地址线 A0 不参加译码，而使A1作最低位线参加译码。可以使用该电路的有效输出去断开含有中断向量的ROM。而在数据总线放上所需的向量，这样重新确定中断服务程序单元。因此外部器件可以规定自己的中断向量。

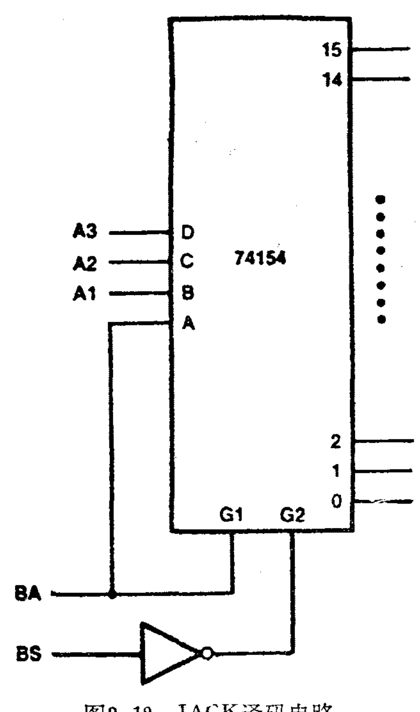


图2.18　IACK译码电路

6809中断向量的存储器地址分配见表2.9所示。

同步确认状态表示6809在程序执行过程中遇有

表 2.9　6809中断向量地址表

向量单元的存储器地址		中断向量说明
高　位	低　位	
FFFE	FFFF	$\overline{RESET}$
FFFC	FFFD	$\overline{NMI}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	保　留

SYNC指令时的输出回答，使6809处在同步状态之下。此时MPU就在等待外部中断线上产生硬件中断信号NMI、FIRQ或IRQ，中断输入信号要保持输入为低电平，特别对FIRQ、IRQ在使用SYNC进行输入判断时，这两种中断信号一定要保持输入的低电平在三个机器周期时

间以上，有关SYNC详情还将在指令系统和硬件接口中介绍。

暂停或总线回答确认状态是表示处理器在$\overline{\text{HALT}}$输入端接受了低电平时，则使正在执行的指令结束后，处理器变为暂停（HALT）状态。在此状态下，处理器停止所有执行过程，而且数据、地址和读/写线都将处在它们的高阻抗状态。因此数据总线和地址总线这时可给外部使用，如直接存储器存取(DMA)。这种状态的存在可以看作是在$\overline{\text{DMA/BREQ}}$或$\overline{\text{HALT}}$线上处在低电平的结果。6809E器件中由于设有$\overline{\text{DMA/BREQ}}$线，不存在总线响应问题。而在6809中设有$\overline{\text{DMA/BREQ}}$线，同时还要使处理器内的全部寄存器的内容自动地进行更新。因此，虽然$\overline{\text{DMA/BREQ}}$输入为低电平，在寄存器进行更新时要使用总线，所以BA、BS不能为1，在自动更新周期内BA、BS暂时表示为低电平。所以在进行$\overline{\text{DMA/BREQ}}$控制时，在DMA执行逻辑中，必须时刻监视BA＝1、BS＝1的状态，即监视BA、BS的信号电平。

（3）R/$\overline{\text{W}}$线

6809的第32线为读/写线也可以认为是状态线。该线和6800的R/$\overline{\text{W}}$线的作用相同，它是表示在数据总线上数据传送的方向。R/$\overline{\text{W}}$＝1时，表示6809在进行读操作；R/$\overline{\text{W}}$＝0时，表示进行写操作，即把数据写到数据总线上。该线为三态信号线，当BA＝1时，R/$\overline{\text{W}}$为高阻抗状态，表示外部可以使用数据总线。R/$\overline{\text{W}}$在Q时钟的上升沿处开始有效。

R/$\overline{\text{W}}$线的另一个用途是确定有效的存储器地址。在6800中使用有效存储器地址线（VMA）表示在地址总线上的有效地址。当在外部译码电路使用该线时，如果在地址总线上的地址无效（VMA＝0），那么所有外部器件都会被禁止工作。6809没有设置VMA线，代替这种作用的是当6809不用数据总线作数据传送时，它将在地址总线上给出FFFF_{16}，并输出R/$\overline{\text{W}}$＝1、BS＝0。因为可以设计外部译码电路来识别上述无效存储器地址状态的条件，这样做可以代替6800的VMA功能。应该注意：无效存储器地址条件与通过总线状态（BS）线取出$\overline{\text{RESET}}$中断向量之间的区别，因为取出向量时BS＝1。

4. 总线定时（E、Q、MRDY）

（1）E、Q定时信号线

E、Q定时信号线分别为34、35引线。这是给6809外部器件提供定时和同步的时钟信号。E时钟信号是6800系列中标准的定时信号，相当于6800的Φ_2时钟信号。当E为高电平时，这就对I/O器件表明地址信息已被放在地址总线上，而且有足够的建立时间，同时数据也被放到

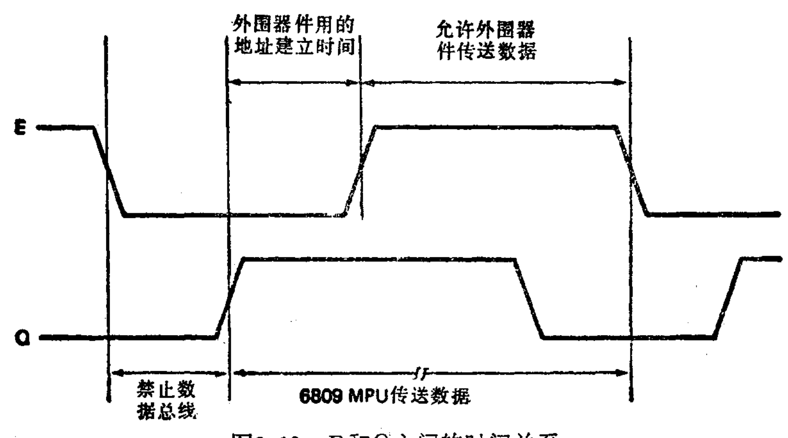
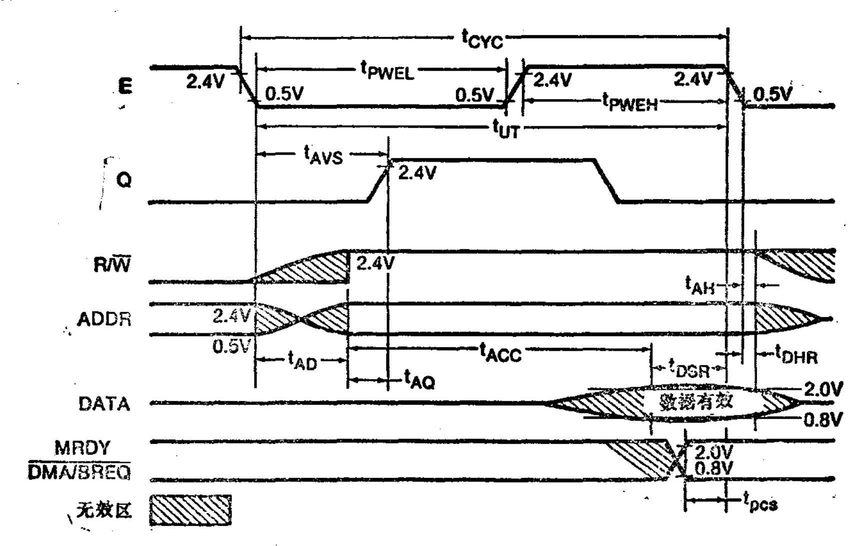
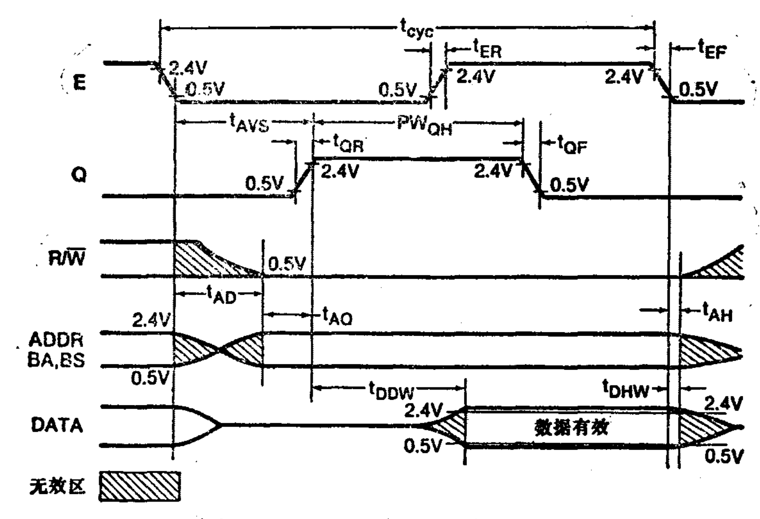


图2.19　E和Q之间的时间关系

数据总线上。该线的信号规定了MPU的周期，而且应该作为任何外部译码机构使用的一部分。Q信号是超前E信号90°的正交时钟信号。E和Q两个时钟信号的关系如图2.19所示。6800系列中没有对应的Q时钟信号。这两个时钟信号的频率等于内部工作频率，但它们之间相移90°，所以给接口线路可提供四个分开的时钟信号沿。6809的读/写时间关系如图2.20所示。从图中可以看到在Q的前沿处地址信息开始有效，在E信号期间数据也有效，而且写入或读出的数据可用E时钟信号的下降沿进行锁存。



（a）从存储器或外部器件读数据



（b）向存储器或外部器件写数据

图2.20　读和写信号的时间关系

根据以上E、Q时钟信号的说明和读/写时间关系图可以得到数据总线和地址总线在使用E、Q时的规则如下：

数据总线规则

• 为使数据总线转向又无竞争时可以用 $\overline{E} \wedge \overline{Q}$ 间隔去控制三态数据总线；

• 在E时钟信号期间，外部器件可以传送数据；

• 除MPU之外，数据总线开始传送数据时可以使用Q时钟信号的前沿；

• 在接收器件中可用E时钟信号的下降沿锁存数据。

地址总线规则

• 在Q时钟信号的上升沿处地址即已稳定；

• 外部器件可用 $\overline{E}\wedge Q$ 作为地址建立时间。

（2）存储器准备好信号（MRDY）

存储器准备好信号是第36线，它和E、Q定时信号一起使用。前面已经说明只要E为高电平时，在数据总线上的数据是有效的。对于某些I/O器件来说，特别是慢速存储器，为满足这些器件的数据的建立、存取和所需要的保持时间，这两种定时信号的时间还不能满足需要。所以采取的办法就是延长E和Q时钟信号时间，以使慢速存储器有时间来响应6809的信号。因此使用MRDY信号线即可实现这种功能。当MRDY为高电平时，E和Q的工作正常。当使MRDY为低电平时，E和Q的时间被延长，数据存取时间即被拉长。E和Q的延长时间将为1/4总线周期的整倍数。只要MRDY保持为低电平，则在MRDY线返回高电平之前，数据有效周期将被延长。然而，最大拉长的周期为10μs（要想采用拉长周期为100μs的6809时，需要特别订货，但价格较贵）。在E和Q时钟信号条件下MRDY的作用如图2.21所示。在存储器的存取无效时（$\overline{VMA}$周期）MRDY也就不会拉长E和Q信号，这样，在不考虑总线存取时，就会防止降低处理器速度。当总线控制已交给外部器件时（在使用HALT和DMA/BREQ情况下），还可以用MRDY来拉长时钟信号（如为慢速存储器使用）。

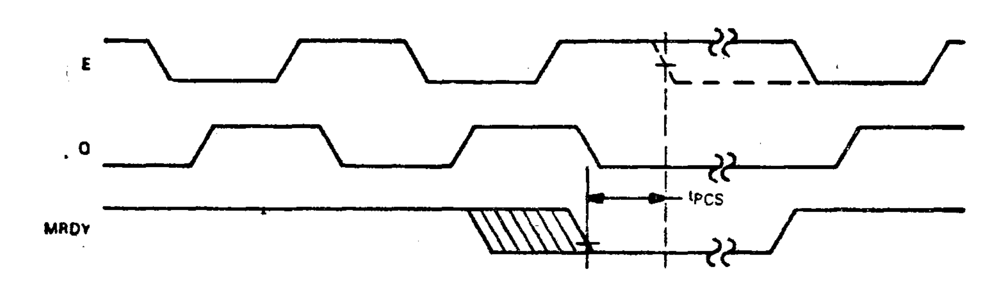


图2.21 MRDY对E和Q的作用

6809早期有四种产品（掩膜型号是G7F、T5A、P6F、T6M），其MRDY的输入需要与4倍频率的时钟进行同步。需要外部振荡器同步的电路如图2.22所示。MRDY信号的负沿通常由选片译码器得来，而且必须符合t_{PCS}定时关系。使用这四种器件时，MRDY的正沿须在4倍时钟频率的上升沿时出现。

另外，这四种器件在$\overline{HALT}$由高到低的时间沿期间，如果机器正在执行TFR或EXG指令时，MRDY就不会拉长E和Q定时信号。如果MPU执行CWAI指令，机器把内部寄存器内容放入堆栈后即等待中断的到来。在等待周期期间，可使MPU进入HALT暂停方式而机器成为三态状态，但这时MRDY不会拉长时钟信号。

以上这四种6809器件在其集成电路封装外壳上面都写有特殊标记。在6809器件名称的下面，前两个字符是该器件掩膜型号代码的后两个字符。后四个数目字表示该器件的出厂日期。这四个数目字代表年和星期。如图2.23所示，是陶瓷封装的T5A掩膜器件，1980年第12个星期制造。如果只有四个数目字即表示为G7F掩膜器件。

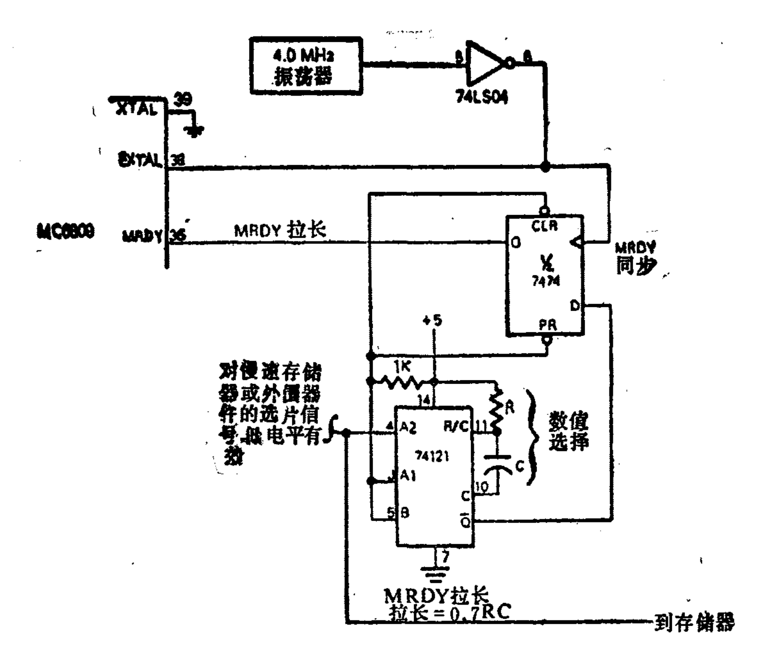


图2.22 MRDY的同步电路

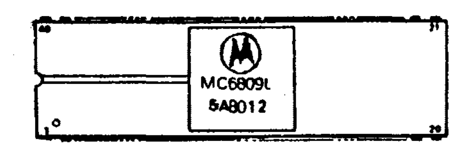


图2.23 T5A型6089器件的出厂标记

5.控制信号线（$\overline{\text{HALT}}$、$\overline{\text{DMA/BREQ}}$、$\overline{\text{RESET}}$、$\overline{\text{NMI}}$、$\overline{\text{IRQ}}$、$\overline{\text{FIRQ}}$）

（1）$\overline{\text{HALT}}$

6809的第40引线是$\overline{\text{HALT}}$，该线提供使6809的操作处于暂停状态。当在该输入端加上低电平时，6809就会在当前指令执行结束后停止运行，而且可长期保持暂停状态，数据永远不丢失。当处于暂停状态时，BA输出为高电平，数据、地址和R/$\overline{\text{W}}$线的三态缓冲驱动器都将变为高阻抗状态，使6809与外部数据和地址总线完全断开。此时BS也将变为高电平，表示处理器处在HALT状态或总线回答状态。在HALT方式时，E和Q总线定时信号会继续正常工作。处在暂停状态时，除可以接受$\overline{\text{DMA/BREQ}}$（下面讨论）外，6809将不响应任何外部控制信号（$\overline{\text{FIRQ}}$、$\overline{\text{IRQ}}$）。然而，如果接收了$\overline{\text{RESET}}$或$\overline{\text{NMI}}$中断时，它们将被锁存起来待暂停终止后再执行。如果MPU因$\overline{\text{RESET}}$或$\overline{\text{DMA/BREQ}}$作用没有运行，就是处在$\overline{\text{RESET}}$为低电平的情况下，只要使$\overline{\text{HALT}}$线变低电平，仍可到达暂停状态（BA・BS＝1）。如果$\overline{\text{DMA/BREQ}}$和$\overline{\text{HALT}}$两者都为低电平，那时当处理器做到指令的最后一个周期之后（与周期窃取相反）机器变为暂停状态。暂停（$\overline{\text{HALT}}$）方式的时间关系如图2.24所示。

$\overline{\text{HALT}}$的功能通常使用在硬件调机和程序调试过程中，因为这种工作方式允许外部装置一次一步地控制程序的执行。同时还可用$\overline{\text{HALT}}$方式作为直接存储器存取工作方式(DMA)，因为这时外部器件可以控制外部总线。

当不使用$\overline{\text{HALT}}$线时，可以使$\overline{\text{HALT}}$线接上直流＋5V电源，以防中断系统的工作。最好串3.3kΩ～4.7kΩ的电阻接到V_{DD}（＋5V）之上。这是因为$\overline{\text{HALT}}$输入端是高阻抗的，开始工作时的状态，不会完全有保证。

（2）$\overline{\text{DMA/BREQ}}$

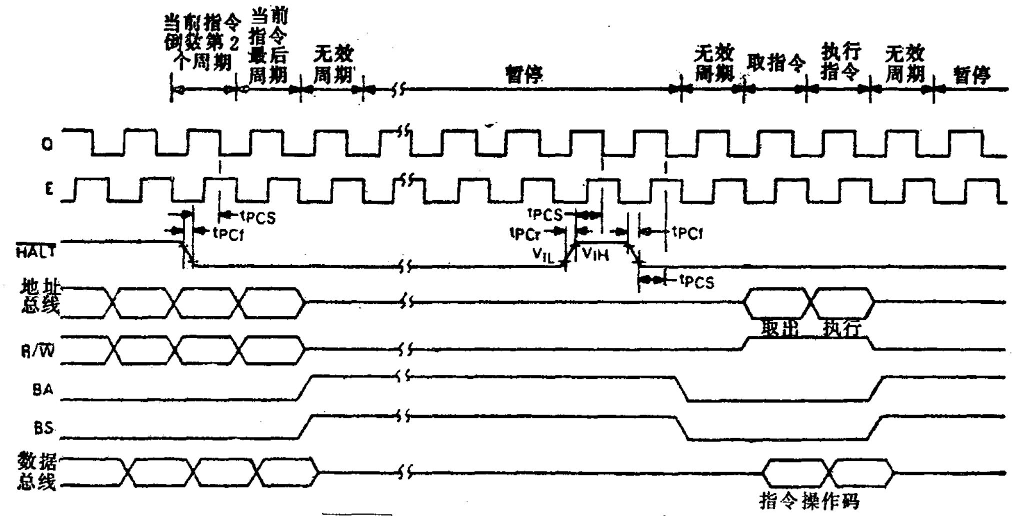


图2.24 $\overline{\text{HALT}}$方式和系统调试中单拍指令的执行过程

6809的第33线是$\overline{\text{DMA/BREQ}}$输入端，该线给外部提供了另一种停止6809执行的方法以及为其它用途得到MPU总线的另一种方式。其时间关系如图2.25所示。该线有两种用途，一是使用DMA方式经总线进行快速存取；二是进行动态存储器的更新。在此对直接存储器存取稍作说明，这种方法的目的是在外部设备和存储器之间不通过MPU而直接进行高速数据传送。通常采用一个单片直接存储器存取控制器（DMAC）来实现直接存储器存取的功能。DMAC器件可以使总线脱离MPU并在存储器和所选的外部设备之间直接进行数据传送。6800系列中的DMA控制器是MC6844。

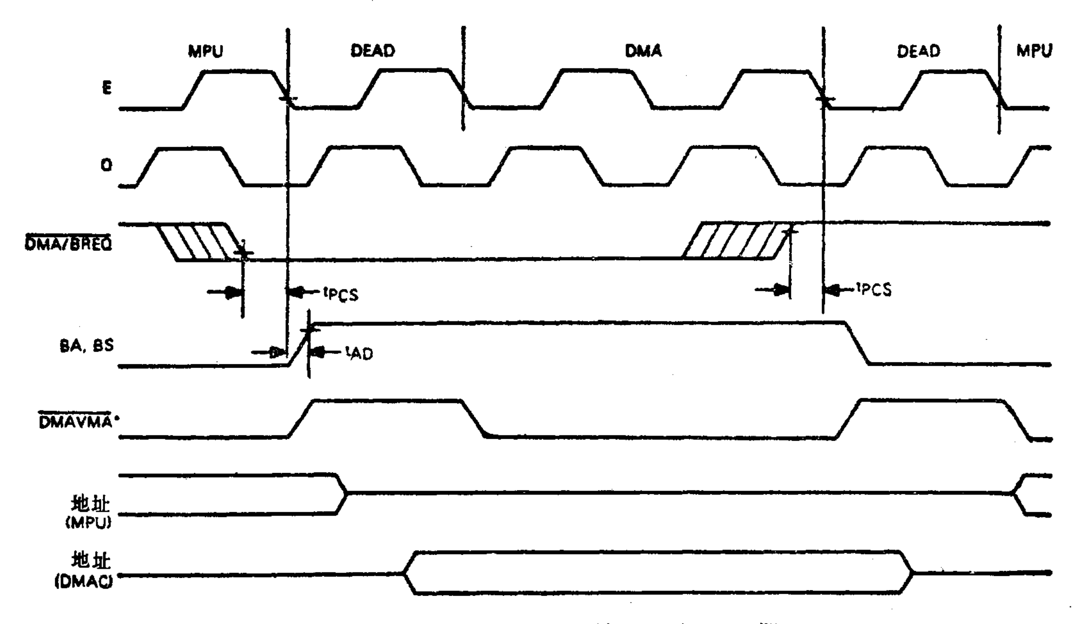


图2.25 DMA方式的时间关系（<14个周期）

从时间关系图中可看到，$\overline{\text{DMA/BREQ}}$的下降沿应在Q信号中间出现，当此时该线为低电平时，除非先由自更新占用之外，MPU就会在当前周期结束时停止指令的执行。此时

MPU将确认$\overline{\mathrm{DMA/BREQ}}$并使BA和BS状态线到高电平。这样，在MPU为自更新寄存器内容而收回总线之前，请求使用总线的外部设备或器件可以有15个总线周期，自更新需要一个总线周期，用死周期[①] 的一个前沿和后沿，见图2.26所示。如果$\overline{\mathrm{DMA/BREQ}}$信号对MPU无作用二个周期时间以上时，那么只是自更新计数器被清除。

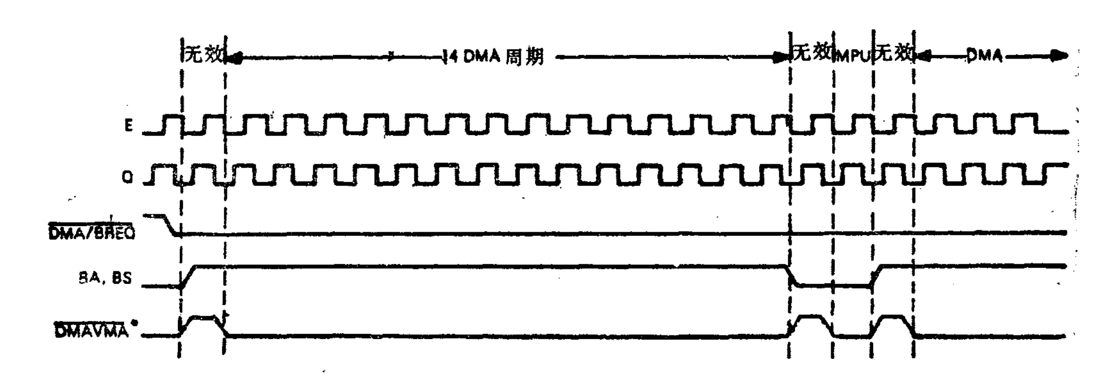


图2.26　自更新DMA时间关系（>14周期）

*　$\overline{\mathrm{DMAVMA}}$是外部构成的信号，是DMA系统要求的

通常DMA控制器要求使用总线时，需要在E信号的前沿即使$\overline{\mathrm{DMA/BREQ}}$保持在低电平上。当MPU使其BA和BS状态输出为高电平作为回答信号时，那么为把总线主权交给DMA控制器时所占用的那个周期即称为死周期。

由于MPU每14个周期要更新一次内部寄存器，对于DMA控制来说要产生三个无效的周期时间，因此需要考虑系统中的DMA控制信号不能处于14个周期以上连续的低电平，这时因为BA、BS变为低电平，DMA或动态存储器的更新必须中断。在无效周期时间内所出现的存储器地址是无效的，因此需要对存储器进行保护。为了防止假存储器存取的发生，在任何死周期之内当BA改变时，在任何周期之中都要产生一个低电平的系统用的$\overline{\mathrm{DMAVMA}}$信号。

当BA变为低电平时（不是因为$\overline{\mathrm{DMA/BREQ}}$为高电平，就是因为MPU自更新），DMA设备应该放弃总线。在MPU访问存储器之前将经过另一个死周期，目的是交出总线主权、避免竞争。

用6809实现DMA控制有三种方法，它们是暂停方式、周期窃取方式和总线多路转接方式。

暂停方式DMA：DMAC使6809$\overline{\mathrm{HALT}}$线变为低电平，并对地址和数据总线加以控制。在数据传送完成之前，6809将一直保持在暂停状态。在数据传送时，总线定时信号E和Q都将继续给出定时信号。

周期窃取DMA：当DMAC使$\overline{\mathrm{DMA/BREQ}}$线变为低电平时，6809将在当前周期结束时停止指令的执行，并使BA和BS总线状态线置为高电平表示确认DMA请求。此时，6809的地址、数据和R/$\overline{\mathrm{W}}$线都将处在高阻抗状态，因此DMAC可以占有对总线结构的控制。在6809用一个MPU周期更新MPU内部寄存器之前，DMAC可有15个MPU周期作数据传送之用。在发生更新周期时，BA线将变为低电平，通知DMAC放弃总线。在一个周期之后，控制将又回交给DMAC，这时$\overline{\mathrm{DMA/BREQ}}$线仍然保持为低电平。这种过程一直被重复进行到$\overline{\mathrm{DMA/BREQ}}$回到高电平为止。关于$\overline{\mathrm{DMA/BREQ}}$发生的顺序如图2.27所示。

---

① 死周期又称无效周期，英文是Dead cycle。

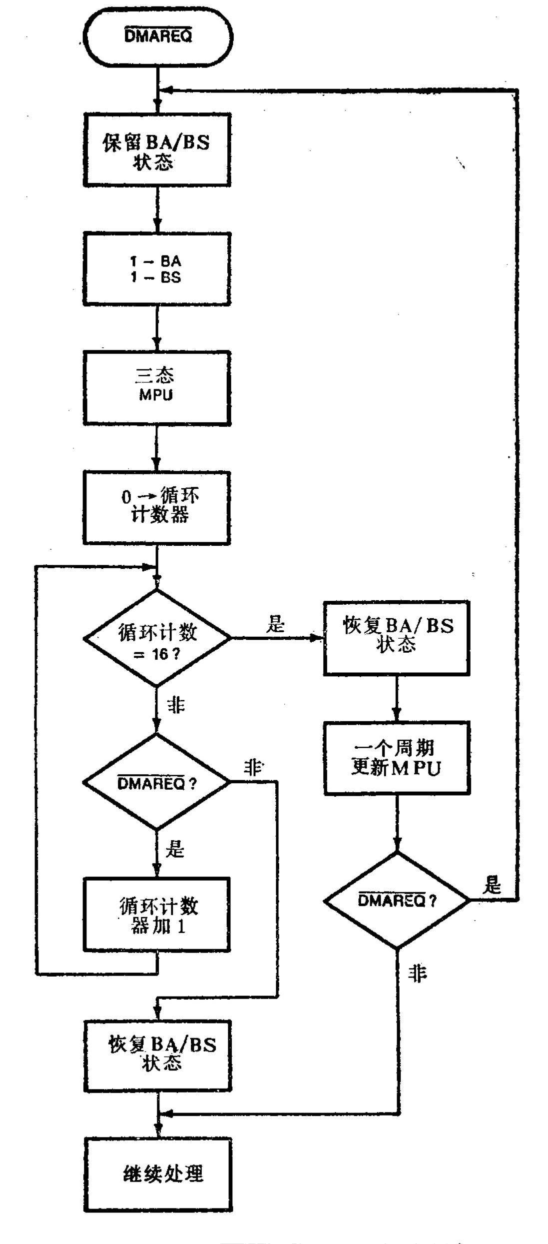


图2.27　$\overline{\text{DMA/BREQ}}$进行的顺序

总线多路转接DMA：当E信号在低电平（此时MPU数据无效）时，DMA外部设备选通控制总线。用这种方法时，6809同DMA外部设备共用总线结构各占50%。这种方法的效率低于其它二种DMA方法，而且还需要更多的外部译码逻辑，所以并不常用。在一般应用中，如象软盘驱动器这种DMA外部设备经常采用DMA暂停方式，而更新动态存储器时常采用周期窃取DMA方式。动态存储器需要每2 ms更新一次，而且优先级很高，在前面$\overline{\text{HALT}}$方式中说过，在此方式时，6809仍会响应$\overline{\text{DMA/BREQ}}$，所以更新存储器不会有问题。

（3）$\overline{\text{RESET}}$

6809第37线为$\overline{\text{RESET}}$输入线，它的作用是初始化或者再启动系统。在该施密特触发器输入端加的低电平信号大于一个总线周期时，即会使MPU总清。其时间关系如图2.28所示。$\overline{\text{RESET}}$信号主要作用是为程序执行提供起始点，这时$\overline{\text{RESET}}$的向量地址从$FFFE_{16}$和$FFFF_{16}$单元中取出（见表2.9），作为将要执行的第一条指令地址放入程序计数器中。系统中为了执行初始化任务需要具有放在ROM中的RESET子程序。在系统加电之后，$\overline{\text{RESET}}$线应该保持低电平一直到内部时钟振荡器完全工作为止再释放。通常保持低电平时间至少20ms，内部时钟才会正常工作。

因为6809 $\overline{\text{RESET}}$输入端设有施密特触发器，它的门限输入电平比6800系列中的外围器件的$\overline{\text{RESET}}$的电平要高，因此采用简单的RC网络即可构成$\overline{\text{RESET}}$电路，总清整个系统。然而，施密特触发器输入至少需要4.0V代表逻辑1的高电平，而其它6809信号可能都采用TTL兼容的最低为2.4V代表逻辑1高电平。在$\overline{\text{RESET}}$输入端采用RC网络的优点在于RC的时间常数，使得在6809完成RESET程序之前，其它外围器件将有足够的时间完成自身RESET，从而开始任何外围设备的初始化程序。

采用电阻和电容的总清电路时，外围器件的$\overline{\text{RESET}}$首先进行总清，然后才是处理器本身的$\overline{\text{RESET}}$进行总清。晶体振荡电路的工作振荡开始之后，到正常状态的时间，如果考虑到各器件$\overline{\text{RESET}}$的总清电平差别时，$\overline{\text{RESET}}$电路时间常数约为0.5s。通常大容量电容器

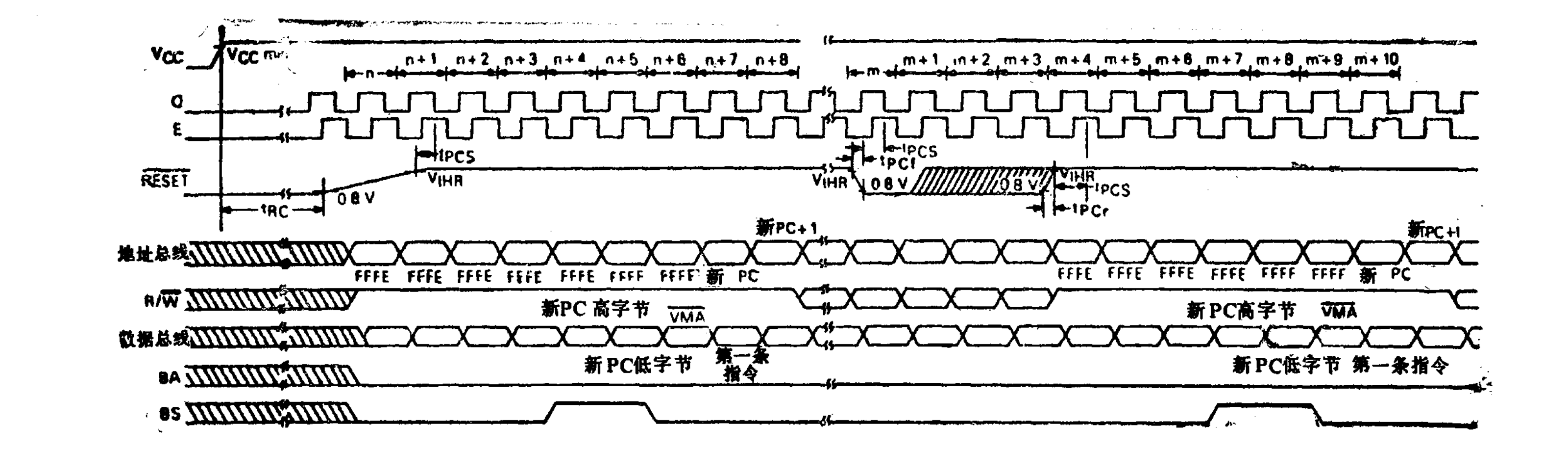


图2.28 $\overline{\text{RESET}}$（总清）时间关系

（100μF）的精度较低，故采用 10kΩ电阻和100μF 电容器作为总清电路，如图2.29所示。

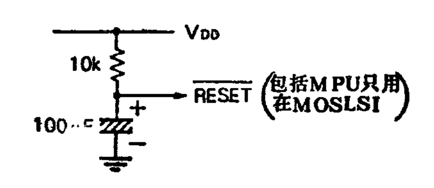


图2.29 $\overline{\text{RESET}}$总清电路

$\overline{\text{RESET}}$过程的处理顺序如图2.30所示。当$\overline{\text{RESET}}$信号至少存在一个MPU周期时才为有效信号，此时当前指令失效退出，而且页面寄存器被清零。接着是所有其它硬件中断($\overline{\text{NMI}}$、$\overline{\text{IRQ}}$、和$\overline{\text{FIRQ}}$) 都被屏蔽或解除。但如果在 $\overline{\text{RESET}}$ 过程中出现了非屏蔽中断$\overline{\text{NMI}}$，则其有效沿将被锁存（保留），而且在任何一种装入S寄存器的指令（LDS、EXG R,S、TFR R,S、LEAS 1，X等）之后都会立即执行$\overline{\text{NMI}}$过程的处理顺序（见图2.32）。能够中断$\overline{\text{RESET}}$处理顺序的外部信号只有$\overline{\text{HALT}}$或$\overline{\text{DMA/BREQ}}$。如果不存在$\overline{\text{HALT}}$或$\overline{\text{DMA/BREQ}}$信号时，总数状态信号BA和 BS 均为低电平，表示MPU处在运行状态。当$\overline{\text{RESET}}$回到高电平（4.0V）时，总线状态信号将表示中断确认（BA＝0，BS＝1）状态。然后，$\overline{\text{RESET}}$的中断向量从地址 FFFE:FFFF 中取出，BS又变为低电平，MPU处在运行状态并执行RESET 中断服务程序。在$\overline{\text{RESET}}$初始化处理期间，如果产生了$\overline{\text{HALT}}$或$\overline{\text{DMA/BREQ}}$，这时就会成为暂停方式或DMA周期窃取方式。在DMA操作期间，如果$\overline{\text{RESET}}$回到高电平，则在DMA操作完成之后，6809就会完成$\overline{\text{RESET}}$处理顺序。最后还应该了解一点，这就是$\overline{\text{HALT}}$或$\overline{\text{DMA/BREQ}}$可以中断 $\overline{\text{RESET}}$ 处理顺序；但$\overline{\text{RESET}}$不能中断$\overline{\text{HALT}}$或$\overline{\text{DMA/BREQ}}$。

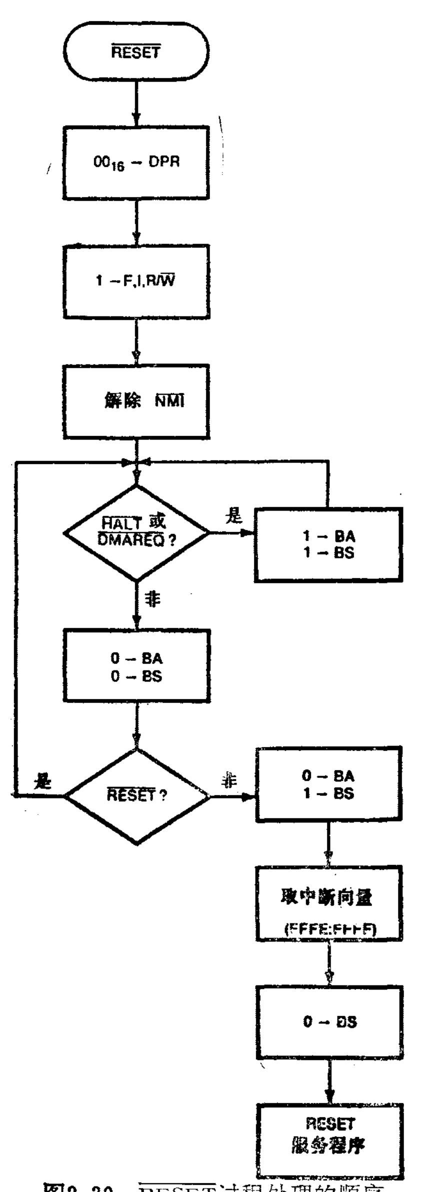


图2.30 $\overline{\text{RESET}}$过程处理的顺序

（4）$\overline{\text{NMI}}$

6809 的非屏蔽中断（$\overline{\text{NMI}}$）非常类似于 6800 的$\overline{\text{NMI}}$。由该中断的名称可知它是不能被程序进行屏蔽的。6809的第 2 线是$\overline{\text{NMI}}$输入端，在该线输入负沿信号时即会产生非屏蔽中断的过程。非屏蔽中断不能被程序禁止，而且其优先级比$\overline{\text{FIRQ}}$、$\overline{\text{IRQ}}$和软件中断要高。在识别$\overline{\text{NMI}}$过程中，全部机器状态都保留在硬件堆栈之中。$\overline{\text{NMI}}$低电平的脉冲宽度至少要有一个E周期的时间。如果输入的 $\overline{\text{NMI}}$ 没有与Q脉冲有最起码的重合，那么在下个周期之前不会识别中断。$\overline{\text{NMI}}$ 的时间关系如图2.31所示。

$\overline{\text{NMI}}$线由于是高阻输入端，所以一定要用 4.7kΩ左右的拉高电阻接到V_{CC}电源上。$\overline{\text{NMI}}$在处理器总清解除之后（即再启动后），到系列堆栈指示器设置数值之前是不能被执行的。

从上节$\overline{\text{RESET}}$讨论中可知在总清处理过程中，禁止$\overline{\text{NMI}}$逻辑发生作用，但在总清操作

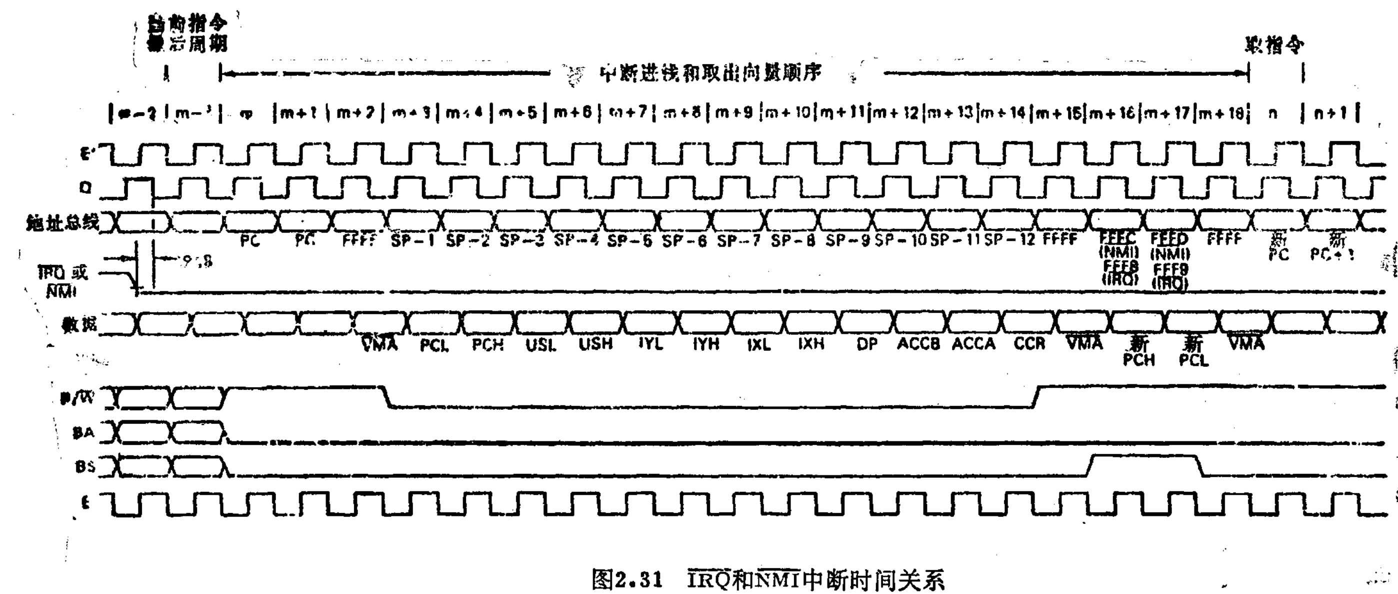


图2.31 $\overline{IRQ}$和$\overline{NMI}$中断时间关系

注：所有输入和输出波形的测量点的逻辑高电平为2.0V，逻辑低电平为0.8V。

期间所出现的$\overline{NMI}$将被锁存下来（保留）只有在S寄存器装入给定数值之后才能$\overline{NMI}$处理顺序。另外该非屏蔽中断如果出现在$\overline{HALT}$操作时，则处理器会把NMI状态保留到MPU脱离$\overline{HALT}$状态之后才可执行$\overline{NMI}$的处理。$\overline{NMI}$的处理顺序如图2.32所示。E标志位置1，表示内部所有寄存器都保留在堆栈之中，然后，按顺序使寄存器进栈。接着使F和I标志位置1，屏蔽中断$\overline{FIRQ}$或$\overline{IRQ}$，这样$\overline{NMI}$执行过程中不接受可屏蔽型中断。但是另一个有效的$\overline{NMI}$可以中断当前在执行的$\overline{NMI}$顺序，因为在$\overline{NMI}$顺序的执行中没有屏蔽$\overline{NMI}$。但如果在该处理顺序完成之前，这种情况连续地重复出现时，那么堆栈就一定会溢出。当$\overline{NMI}$中断向量从地址FFFC：FFFD中取出时，总线状态线(BA/BS)即表示中断确认状态（IACK）。等到把取出的向量装到程序计数器之后，则总线状态线BS变为低电平，MPU处在正常运行状态，并执行$\overline{NMI}$中断服务程序。当完成中断服务程序时，程序一定回到主程序。这时应在中断服务程序中的最后一条指令放入中断返回指令RTI。RTI指令可使6809回到原来寄存器没有进栈时的状态。

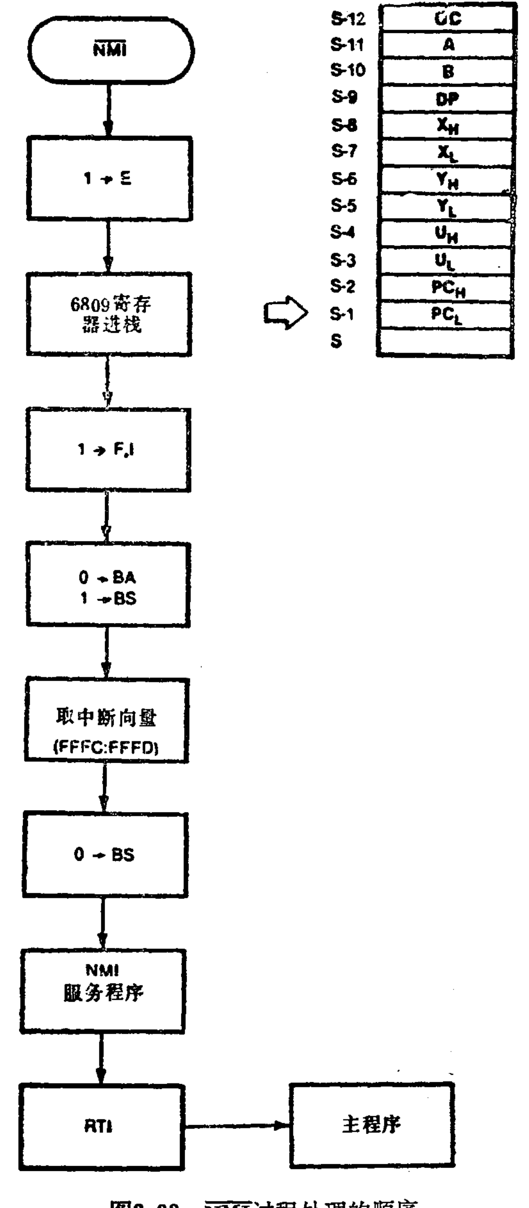


图2.32　$\overline{NMI}$过程处理的顺序

（5）$\overline{IRQ}$

6809的第3线是中断请求输入端。当CCR中的屏蔽位I为0，而该输入端又为低电平时，则会开始中断请求过程。因为$\overline{IRQ}$时使MPU的全部状态都保留在堆栈之中，所以比$\overline{FIRQ}$的中断响应速度慢。$\overline{IRQ}$的优先级也比$\overline{FIRQ}$低。还有一点要说明的就是$\overline{IRQ}$的中断服务程序之中，在执行RTI指令之前，应该使中断源清除。$\overline{IRQ}$的时间关系如图2.31所示。

$\overline{IRQ}$处理过程的顺序如图2.33所示。使条件码寄存器中的I位置1，即可屏蔽$\overline{IRQ}$发生。如果I位为0，则可接受中断，接受中断之后，$\overline{IRQ}$的处理顺序和$\overline{NMI}$的过程类似。使E标志位置1，全部内部寄存器内容按顺序放入S堆栈，然后使I位置1，以便在当前中断执行完之前屏蔽任何其它$\overline{IRQ}$中断。但应注意之点是，F标志位并没有置1，所以$\overline{IRQ}$的处理顺序可以被快速中断请求$\overline{FIRQ}$进行中断。因此可以说$\overline{FIRQ}$的优先级高于$\overline{IRQ}$。当$\overline{IRQ}$中断向量从地址FFF8：FFF9中取出时，总线状态线表示MPU的中断确认状态（IACK）。一旦取出了向量之后，总线状态线BS变为低电平，MPU处于正常运行状态，执行$\overline{IRQ}$中断服务

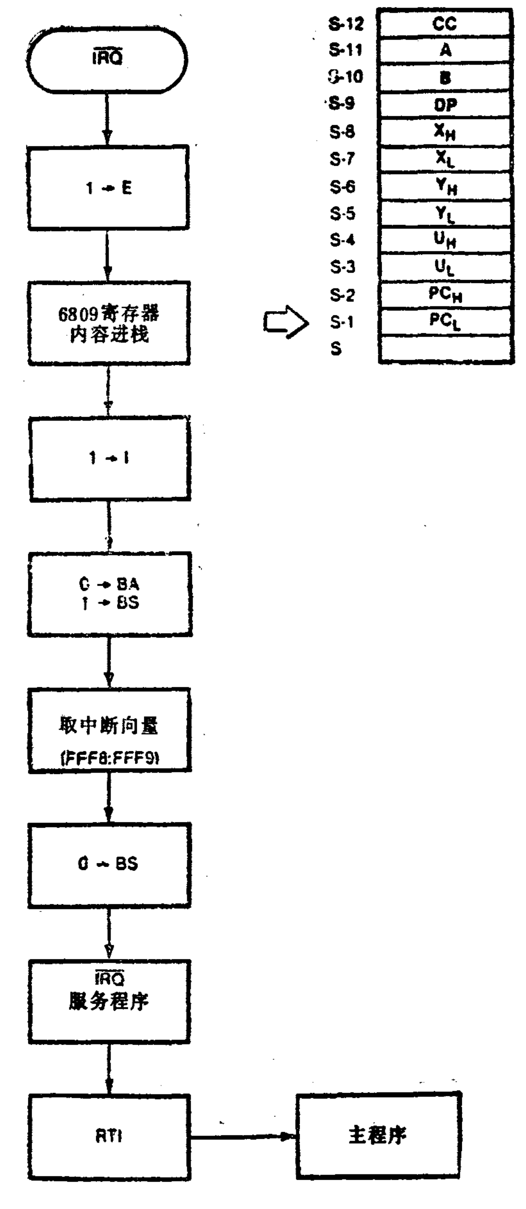


图2.33　$\overline{\text{IRQ}}$过程处理的顺序

程序。再一点就是，在中断服务程序的最后要用RTI指令返回到主程序。

（6）$\overline{\text{FIRQ}}$

6809的第4线是快速中断请求输入端。当CCR中的屏蔽位F为0，而该输入端又为低电平时，则会开始快速中断请求过程。$\overline{\text{FIRQ}}$中断和一般的中断不同。其优先级比$\overline{\text{IRQ}}$高，而且因为在中断时只有条件码寄存器和程序计数器的内容保留在堆栈之中，所以中断的响应是快速的。在快速中断处理程序中需要再保留的其它寄存器可在中断程序中进行。在执行中断返回指令RTI之前，中断服务程序应该使中断源清零。其输入电路也为高阻抗电路，因为是靠电平检出后工作，所以在执行快速中断之前，中断信号要保持一定宽度的电平。$\overline{\text{FIRQ}}$的时间关系如图2.34所示。

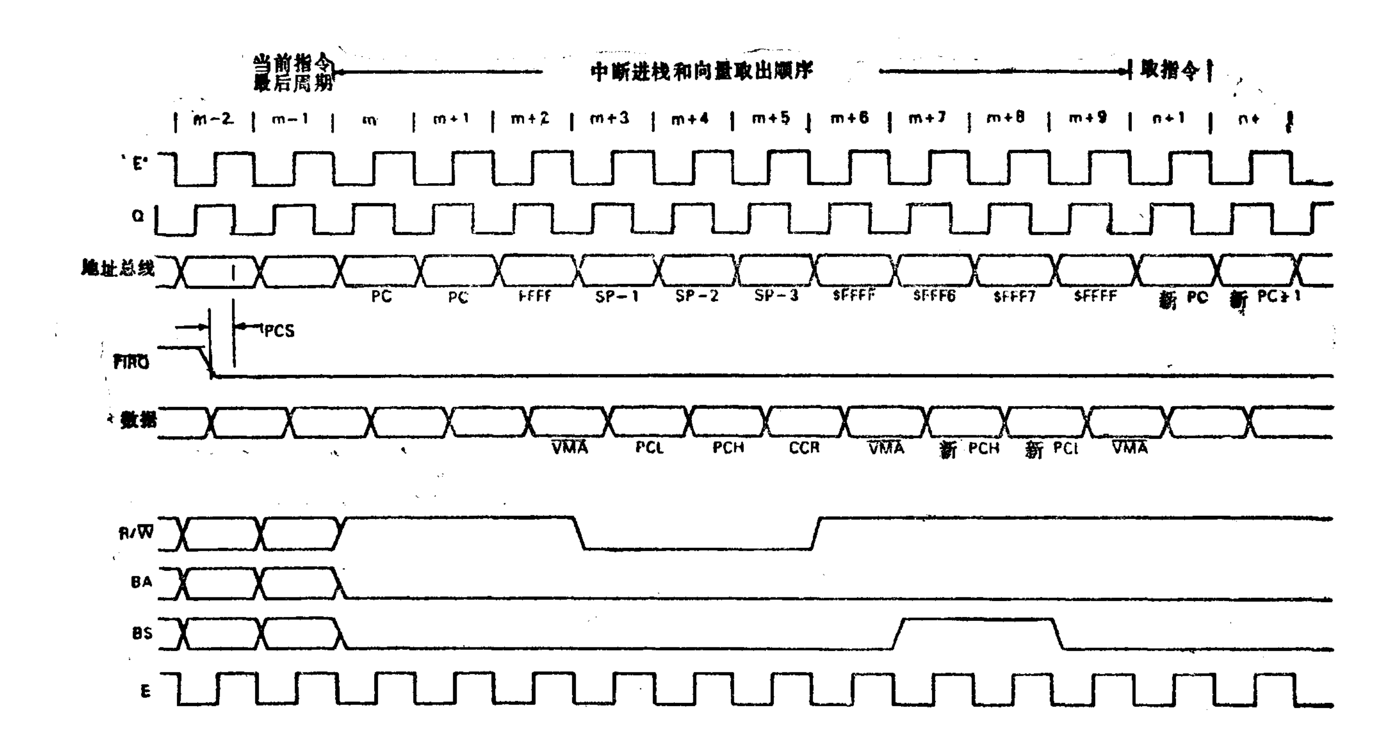


图2.34 $\overline{\text{FIRQ}}$中断时间关系

注：所有输入和输出波形测量点的逻辑高电平为2.0V，逻辑低电平为0.8V

$\overline{\text{FIRQ}}$过程的处理顺序如图2.35所示。应该注意其与$\overline{\text{IRQ}}$（图2.33）处理顺序的差别。E标志位被清零是表示只有程序计数器和条件码寄存器存入堆栈。F和I位都被置1，说明该处理过程中不要被任何$\overline{\text{IRQ}}$或其它的$\overline{\text{FIRQ}}$中断。但如果不希望优先权自动按$\overline{\text{FIRQ}}$/$\overline{\text{IRQ}}$顺序，则在中断处理程序中I位可以清零。除去$\overline{\text{FIRQ}}$中断向量地址在FFF 6：FFF 7之外，其余的处理过程都和$\overline{\text{IRQ}}$的过程一样。如上所说，$\overline{\text{FIRQ}}$比$\overline{\text{IRQ}}$的优点就是在中断时不是所有内部寄存器的内容都被保留，因此节约中断响应时间。如需增多保留寄存器内容时，可在$\overline{\text{FIRQ}}$中断服务程序中使用PSHS/PULS指令进行。

在使用MRDY或者$\overline{\text{DMA/BREQ}}$控制内部MPU时钟时，一定要注意在MRDY或$\overline{\text{DMA}}$/

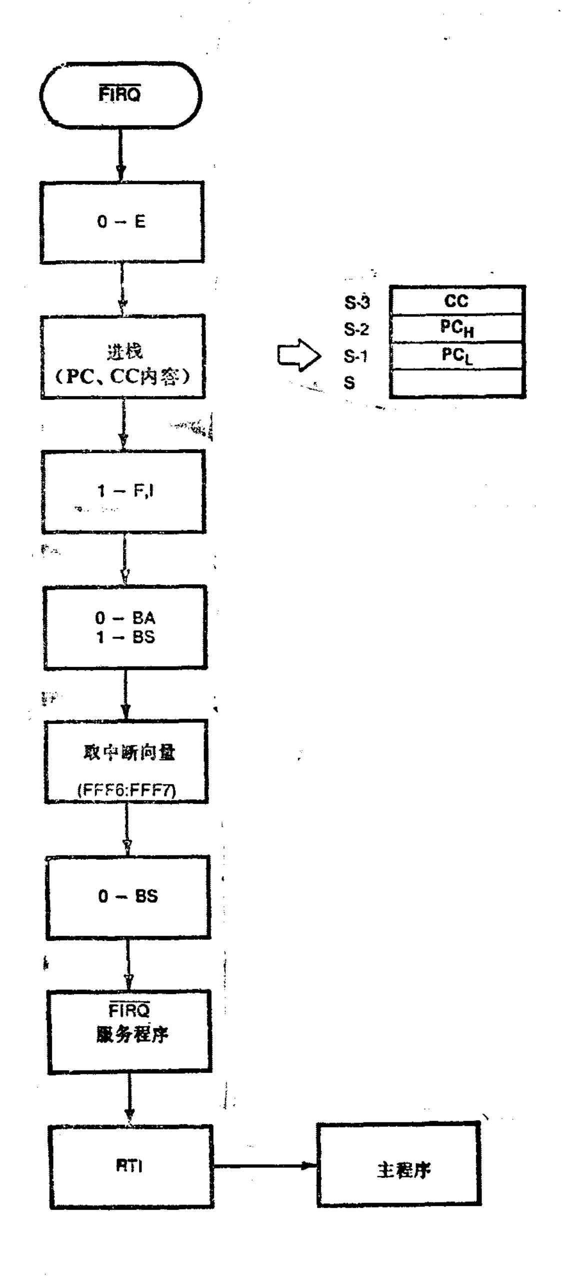


图2.35　$\overline{\text{FIRQ}}$过程的处理顺序

$\overline{BREQ}$期间不能使t_{cyc}的周期时间超过16μs。

$\overline{NMI}$、$\overline{FIRQ}$和$\overline{IRQ}$要求在Q的下降沿采样，在识别这些中断之前，需要有一个周期时间进行同步。当前指令完成之前中断不能进行服务，除非有SYNC或CWAI条件出现。如在当前指令完成时，$\overline{IRQ}$和$\overline{FIRQ}$不为低电平，显然不能识别中断。但$\overline{NMI}$只要保持低电平一个周期时间即可被锁存。在$\overline{RESET}$的下降沿和表示$\overline{RESET}$确认的BS的上升沿之间都不能识别或锁存中断。

控制信号线小结

控制信号	重要性能
$\overline{RESET}$	1. 在MPU之前，先使每个外部器件总清（不同于6800）
	2. 对整个系统可简单使用RC电路
$\overline{HALT}$	1. 可用于周期窃取、成组DMA、或单拍指令执行
	2. 在当前指令完成后实行
$\overline{DMA/BREQ}$	1. 加速存储器存取或多处理技术应用
	2. 给用户使用15个连续总线周期
	3. 在一个时钟周期之内得到总线
$\overline{NMI}$	1. 保留全部机器状态
	2. 一定至少保持一个E周期宽度的低电平
	3. 在$\overline{RESET}$之后，只有第一条程序是装入硬件堆栈指示器内容才识别$\overline{NMI}$
$\overline{FIRQ}$	1. 只使CCR和PC进入堆栈
	2. CCR中F位为0，才会识别该中断
$\overline{IRQ}$	1. CCR中I位为0，才会识别该中断
	2. 所有寄存器保留在堆栈

2.3.2 6809E的引线

6809E是6809的姐妹处理器，它也是一种40条引线的HMOS器件，有塑料封装（P）和陶瓷封装（L）两种外壳。设计该器件的主要目的是为多处理器系统使用，因为它增多了MPU状态信息用的外部信号线。6809E的引线配置情况如图2.36所示。新增加的两条外部信

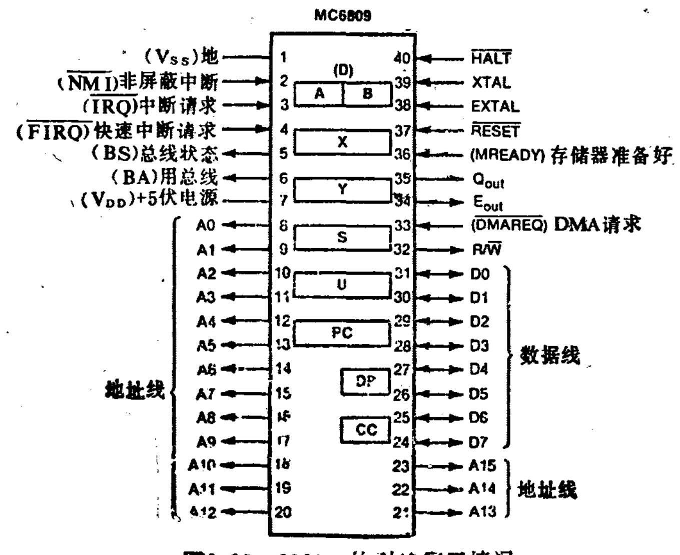
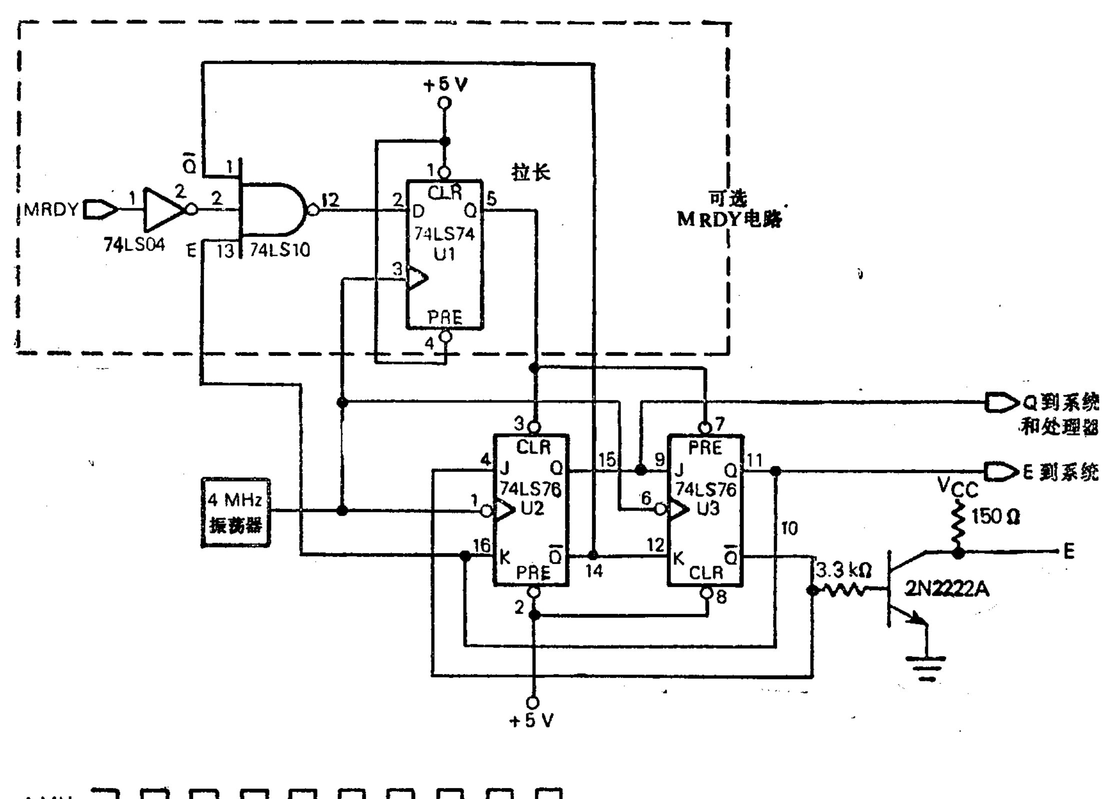


图2.36 6809E的引线配置情况

号线（BUSY和LIC）代替了6809的两条晶体连接线（XTAL和EXTAL）。而且6809E又设置一条先行有效存储器地址线（AVMA）代替6809存储器准备好线（MRDY），同时还设置了一条三态控制线（TSC）代替6809的$\overline{DMA/BREQ}$线。因此，本节主要说明6809E的特殊引线，和6809相同的引线就不再赘述。

1. 时钟输入线（E、Q）

6809E型器件内部没有设置时钟振荡器，它必须由外部时钟分别经过第34线和第35线作为E和Q时钟脉冲输入信号。6809E时钟E、Q间的相位关系和6809的E、Q关系相同。Q必须超前E，这就是说Q的脉冲沿后面必须跟有最低限度延迟的同样的E脉冲沿。MPU的输出地址是在E的下降沿后经过 t_{AD} 时间有效；总线上的数据锁存也用E脉冲的下降沿进行。Q输入端完全和TTL兼容；然而，E输入端因为是直接驱动内部MOS电路，所以需要一种高于标准TTL的电平。6809E的时钟振荡器电路和E、Q时间关系如图2.37所示。电路图中的E和Q输



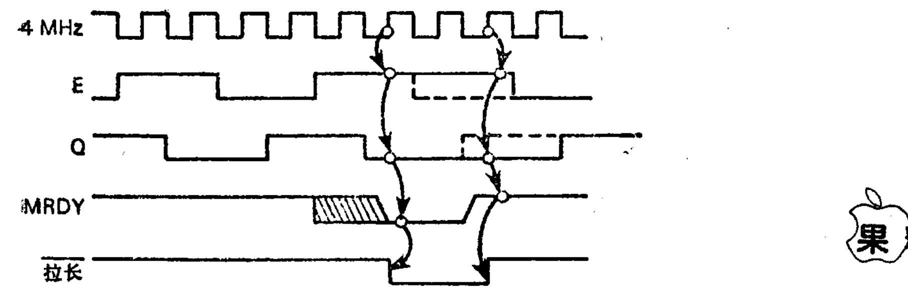


图2.37 6809E 时钟产生器和时间关系

注：如果无可选电路，U2和U3的CLR，PRE端须接高电平

出端可以直接分别加到6809E的34和35线上，同时该电路还为外部电路定时提供适当的TTL信号。其中振荡器频率是所要求时钟频率f_c的4倍。由该电路所产生的时钟频率直接驱动6809E，不用6809E的内部逻辑除以4。6809E的标准工作频率为1MHz，还有工作频率为1.5MHz的68A09E和2.0MHz的68B09E。

2. 状态线（BUSY、LIC）

6809E又多设两条状态输出线如下：

（1）BUSY

6809E的第33线为BUSY输出线，该线的状态表示6809E正在访问存储器。BUSY输出端只在使用外部时钟的6809E中存在，而在内含时钟的6809之中，不能用外部电路来设置BUSY信号。在读-修改-写存储器指令（如ASL等指令）执行期间，或象对LDD、STD、ADDD这种两字节数据在第一个字节存取期间，BUSY信号输出将处于高电平。但对PSHU、PULU、PSHS、PULS指令的动作，虽然操作为二字节数据，BUSY信号并不处于高电平。同时，在任何间址中的第一个字节或其它向量取出期间（如扩充JUMP、SWI间址等），BUSY也为高电平。

在多处理器系统中，执行读-修改-写过程时，在读之后而写执行结束之前，BUSY信号输出可以禁止其它处理器错误地读出原来的数据；而且利用该线在整个读/写周期为高电平、周期结束时为低电平的原理，对系统的工作进行监视，所以在一定的时间之内，只有一个处理器去访问公用存储器或外部设备。同时，在多处理器系统中，BUSY信号还可以表示对下一个总线周期延期重新仲裁的要求，以便确保上述操作的完整性。其区别就在于使用任何一种读-修改-写指令时，都需要对"测试并置位"的基元操作提供不可分割的存储器存取动作。

典型的读-修改-写指令（ASL）的操作示于图2.38。其操作过程的时间关系如图2.39所示。在Q的上升沿之后的控制延迟时间t_{CD}处BUSY成为有效。

在多处理器系统中，为了解译各种MPU的状态信号和仲裁各处理器之间共用的存储器或外部设备，需要使用BUSY作为外部总线仲裁逻辑的控制信号。

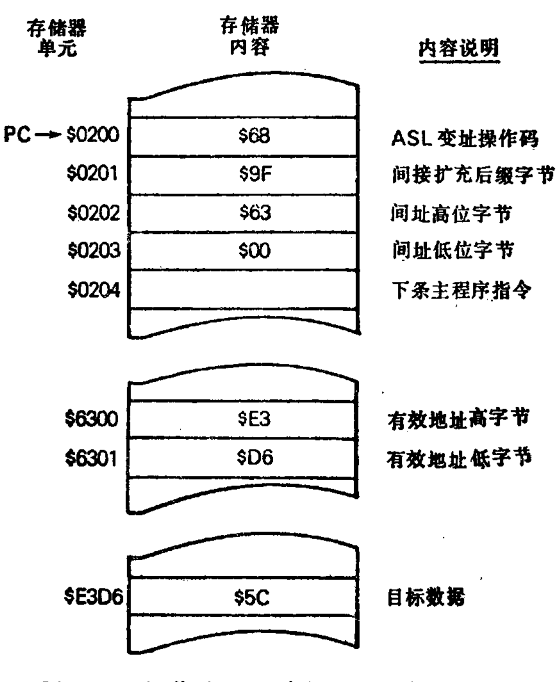


图2.38 读-修改-写指令例（ASL扩充间址）

（2）LIC

6809E的第38线为最后指令周期（LIC）的状态输出线。在每条指令执行的最后周期时，LIC变为高电平，因此LIC处在高电平时，表示下个机器周期为取出操作码；而且LIC从高到低电平的下降沿表示操作码的第一个字节在当前总线周期结束时将被锁存。当MPU在指令结束被暂停执行时（即不在CWAI或$\overline{RESET}$时），在同步状态、或者在中断期间进行存栈时，LIC都将为高电平状态。在Q的上升沿后延迟t_{CD}时间LIC有效。关于LIC的时间关系

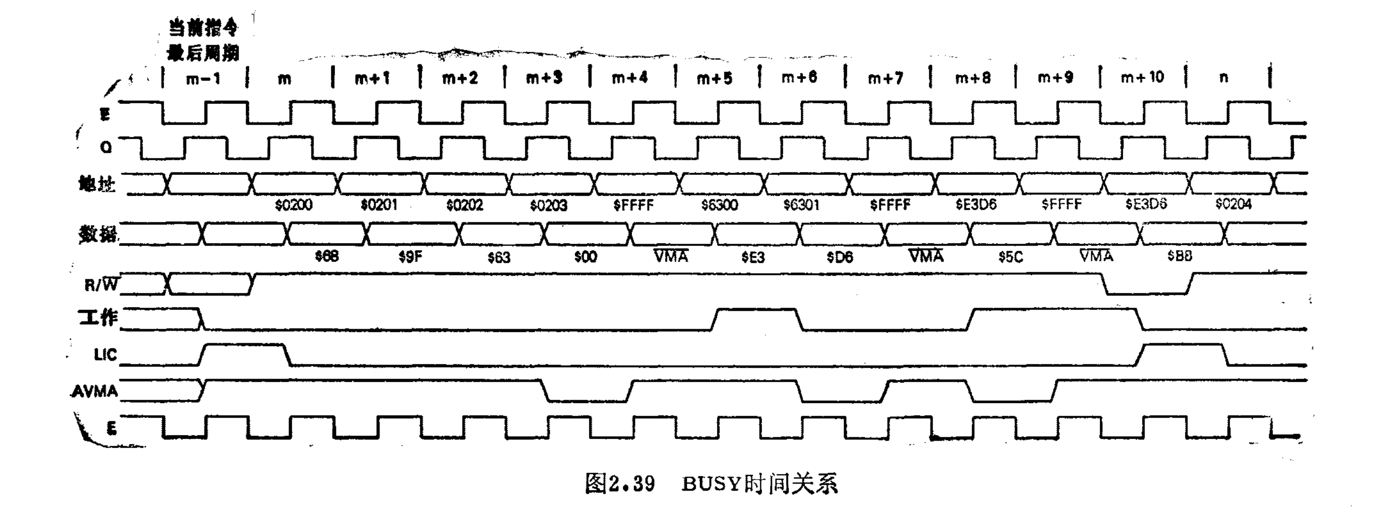


图2.39 BUSY时间关系

见图2.39所示。

LIC的状态线可以在和6809E相互同步工作的协同处理器中作为控制信号使用。

协同处理器和多处理器的区别是：多处理器系统中，在各处理器之间，其指令的取出和执行是无须进行同步的；但在协同处理器中，辅助的协同处理器和6809E处理器中指令的取出和执行都要同步。例如为了增强6809E的运算功能，可采用具有浮点运算的LSI单片机作为协同处理器。而在多处理器系统中，各处理器都设有自己专用的存储器和外围器件，只有在存取共享资源时其影响才存在。协同处理器是多处理器系统中的一种形式，硬件上结合得更为密切。

可以使用LIC线通知多处理器总线仲裁逻辑：MPU准备完成自己的指令周期，所以开始启动仲裁过程。

3. 控制线（AVMA、TSC）

（1）AVMA

6809E的第36线是先行有效存储器地址输出线。AVMA线相当于6800系列中设置的VMA输出线。但和VMA的不同之处是：不仅AVMA处在高电平的周期时，地址总线输出有效；而且在下一个周期的地址总线输出也是有效的。这就是说，当6809E准备使用数据/地址总线时，该线即变为逻辑1态。而且该输出从低电平到高电平时，即表示MPU在下一个总线周期之中将使用总线。AVMA信号的这种预测性能非常有效地适用于共享总线的多处理器系统，在这种共享总线系统中，控制总线主权的设备在经常改变，因此有的设备必须放弃总线主权。当MPU处在$\overline{\text{HALT}}$或SYNC状态时，AVMA为低电平。在Q上升沿之后延迟t_{CD}时AVMA有效。

AVMA早输出一个机器周期的理由是：在多处理器运行过程中，通常需要在下一个周期使用大容量存储器或外部设备等共享资源，如若共享资源为BUSY（使用）状态，那么就需要延长E、Q脉冲，等待共享资源可以进行存取，因此共享资源本身应该使用更高速的逻辑电路。

为了形成和6800系列VMA同样的输出，可以使用D型触发器实现，如图2.40所示。

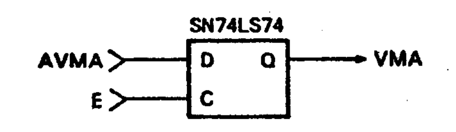


图2.40 VMA的形成方法

（2）TSC

6809E的第39线为三态控制输入线（TSC），该线类似于6809的$\overline{\text{DMA/BREQ}}$控制线。当TSC线输入为高电平时，将使6809E的MOS型地址、数据和R/$\overline{\text{W}}$缓冲驱动器处于高阻抗状态。而控制信号（BA、BS、BUSY、AVMA和LIC）并不处在高阻抗状态。

设置TSC输入的目的在于其它总线控制器（处理器或DMA控制器）可以共享一条总线。所以它是从外部直接使数据总线、地址总线和R/$\overline{\text{W}}$信号同处理器相脱离的信号。在共用总线的多处理器系统或组成的DMA系统中，在不完全降低处理器处理能力的条件下，使用TSC是密耦合多机处理的一种方式。

当TSC为高电平时，MPU输入时钟信号E、Q一定得停止，而当TSC回到低电平时，时钟信号则被重新启动。所以可以用TSC提供DMA的周期窃取方式或动态存储器更新。

TSC信号加入后的时间关系如图2.41所示。当E为低电平、或者从高电平变为低电平

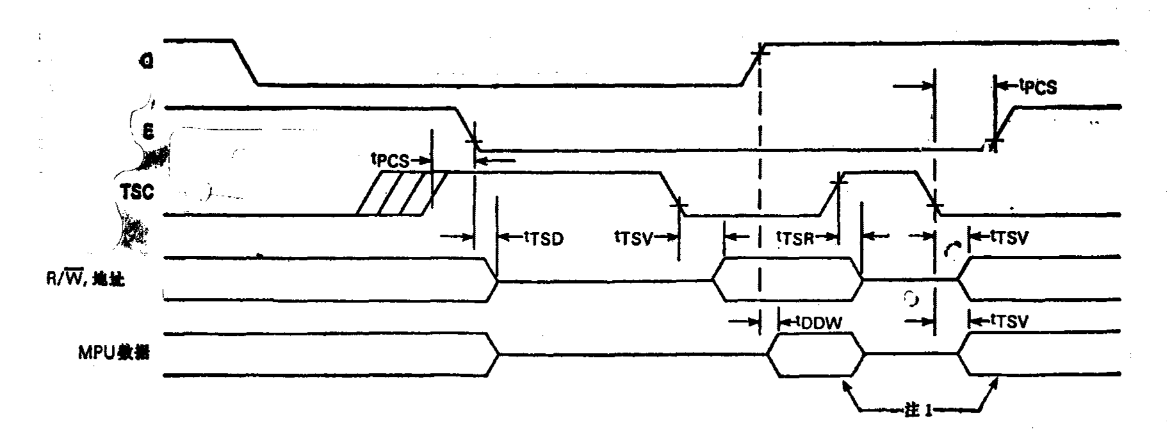


图2.41 TSC的时间关系

注 1：只有在R/W̄为低电平，而且E或Q为高电平期间，MPU才确定数据。为说明多数定时情况，给出合成总线周期

注 2：时间测量在低电平0.8v，高电平2.0v 处进行，否则另有说明

时，地址总线和R/W̄线、以及在写操作期间的数据总线都将处在高阻抗状态，直到Q脉冲上升沿为止；但如果在此时间TSC仍为高电平，则上述这些总线将继续保持在高阻抗状态。如果TSC的高电平时间超过E的上升沿时间，那么它将被内部锁存，并在其余的总线周期总线驱动器仍处在高阻抗状态。所以，在利用TSC的系统中，E为高电平期间，处理器用；E为低电平期间DMA或其它处理器用，故它可以应用在双处理器系统的切换之中。

4. 6809E的其它引线

上述说明的6809E的引线主要是与6809相区别的输入/输出线。至于和6809相同的引线其电气特性和逻辑功能基本上和6809的一样。这些信号线的时间关系图见附录。

2.3.3 6809/6809E的输入输出电路

1. 输入电路

输入电路的工作电平，全都和TTL电路相兼容。但是$\overline{\text{RESET}}$信号比外围芯片上的$\overline{\text{RESET}}$所需的TTL电平要有较高的电平值。

各个输入端的输入电容量，除去采用外部时钟的6809E中的E、Q端之外最大为10pF。而6809E型的E、Q脉冲输入端要求E、Q脉冲在芯片内高速工作，所以上升沿时间要求极快。因而如图2.42所示的输入保护电路和地址总线、数据总线上的不同，在E、Q脉冲输入端要使用特别小的扩散电阻。

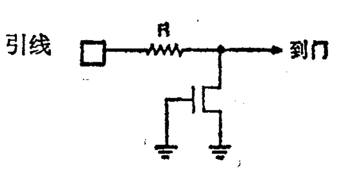


图2.42 输入端保护电路

还有，在6809型处理器中，为了用EXTAL构成晶体振荡电路，也不能使用很大的扩散电阻，所以和6809E的E、Q端一样应为较小的数值。为了防止由于静电造成的损坏，在所有输入端上都要加输入保护电路，对于EXTAL（指6809）端、XTAL（指6809E）端、Q端（指6809E）需特别加小的保护电阻，要充分注意防止静电致损。

2. 中断输入（$\overline{\text{FIRQ}}$、$\overline{\text{IRQ}}$、$\overline{\text{NMI}}$）电路中注意之点

当使用外围芯片上的$\overline{\text{IRQ}}$作输出驱动器时为了形成“线或”能力，在其V_{DD}端上都还没有

接上电路，因此要在处理器的输入端和 V_{DD} 之间必须接入 3.3～4.7kΩ 的拉高电阻。如果没有拉高电阻，处理器将会经常陷入中断状态，完全不能工作。在系统中的中断输入电路如图2.43所示。

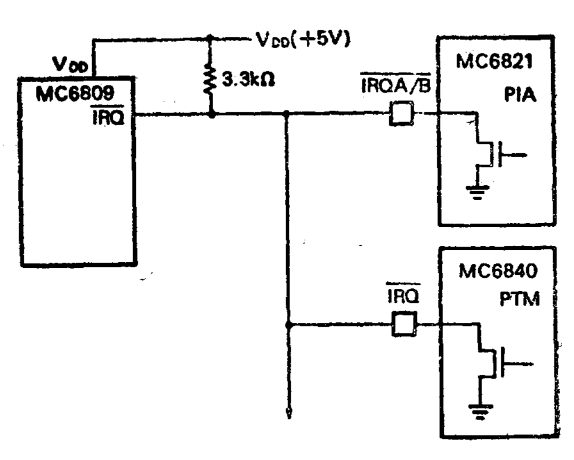
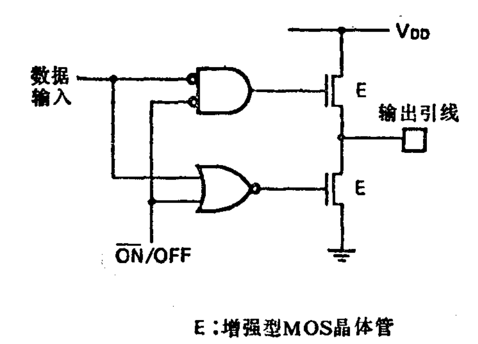


图2.43 中断输入端电路

3. 输出电路

输出电路有地址总线驱动器、数据总线驱动器和总线状态输出电路三种。地址总线驱动器由高速推挽电路构成，也可成为高阻抗状态，推挽驱动器是按可以驱动一个肖特基TTL电路或 4 个低功耗的LS型 TTL 电路设计的。同时对 MOS 负载电容量要求：在 $R/\overline{W}$、地址总线驱动器输出端可驱动90pF，在数据总线驱动器输出端可驱动130pF。输出基本电路如图2.44所示。



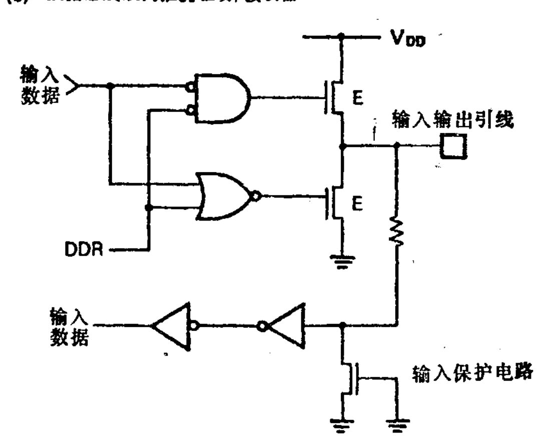


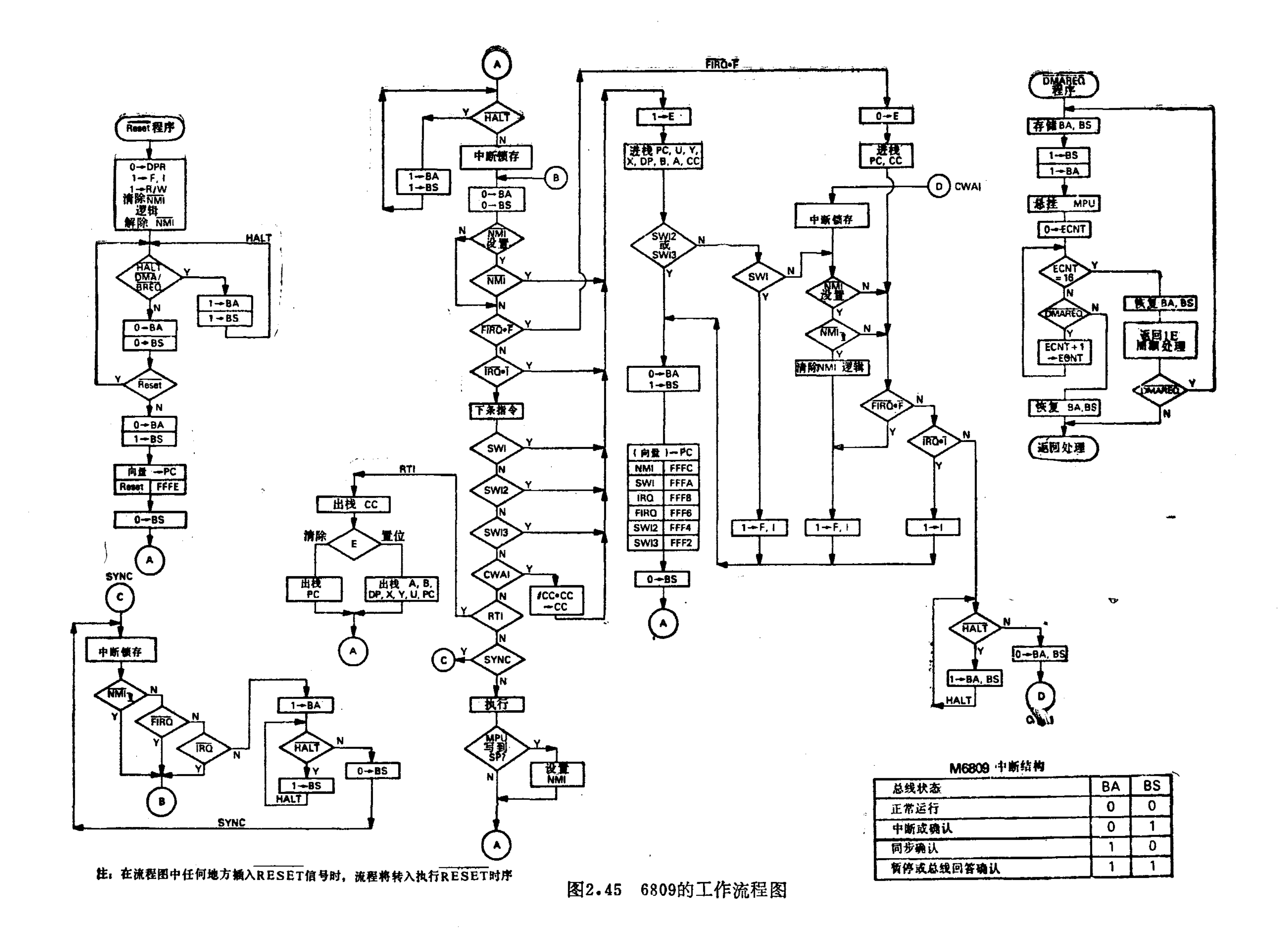
图2.44 输出基本电路

6809系统中完全不用总线缓冲驱动器时，可以并联接在总线上的6800系列外围大规模电路芯片可达 8 个。

2.4 6809的工作原理概述

6809的工作流程图如图2.45所示。

MPU的系统加了电源以后，或是按动了手动开关之后，$\overline{RESET}$输入端如果从低电平变到高电平，经过 8 个机器周期以上则完成总清（复位）状态。其过程也可称为再启动过程，如图2.46所示，详细动作过程见图2.45有关内容。经启动过程之后，在PC计数器的高低字节中分别装入了FFFE和FFFF向量单元中的内容作为程序的起始地址。此后，$\overline{HALT}$ 输入如果是低电平，则暂停；如果为高电平，当有中断输入时，就进行中断处理，没有中断输入时，就取指令。取出的指令如果不是软中断指令，也不是CWAI、RTI或SYNC指令时，就



M6809 中断结构

总线状态	BA	BS
正常运行	0	0
中断或确认	0	1
同步确认	1	0
暂停或总线回答确认	1	1

注：在流程图中任何地方插入$\overline{\text{RESET}}$信号时，流程将转入执行$\overline{\text{RESET}}$时序

图2.45 6809的工作流程图

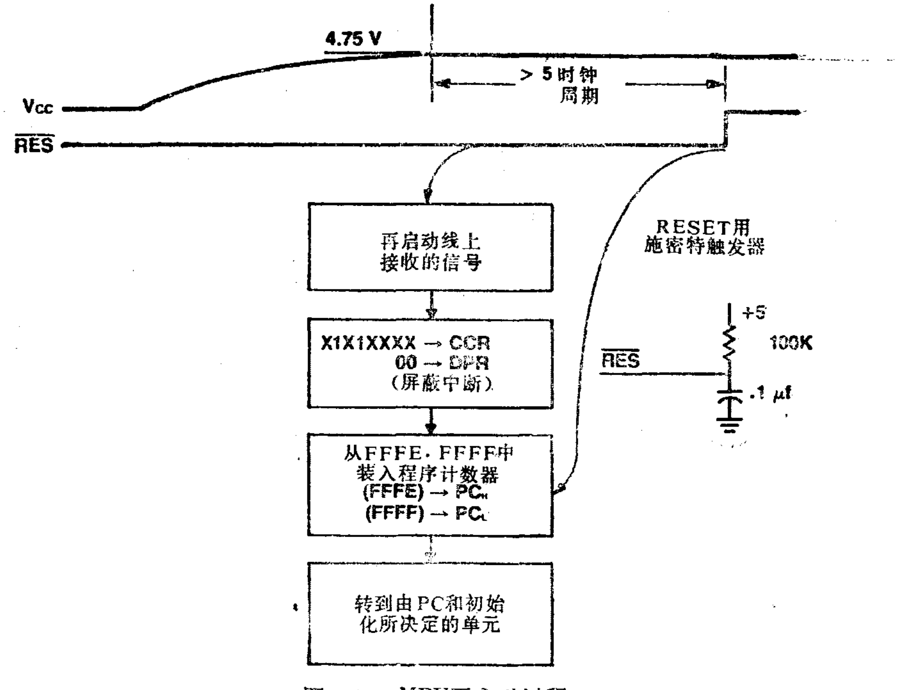


图2.46 MPU再启动过程

继续执行指令。在正常工作期间，MPU从存储器中取出指令，然后执行指令所要求的功能。除非出现了特殊指令，如SWI、SWI 2、SWI 3、CWAI、RTI和SYNC或者硬件有新的情况发生，可改变MPU正常工作过程之外，这种过程会无限重复下去。

6809的中断、$\overline{\text{HALT}}$或$\overline{\text{DMA/BREQ}}$也能改变正常的指令执行顺序。如前所述6809有$\overline{\text{NMI}}$、$\overline{\text{FIRQ}}$、$\overline{\text{IRQ}}$三级硬件中断，和SWI、SMI 2、SWI 3三种软件中断，为了更详细了解其相互间级别的关系和总线状态，从6809MPU的流程图说明是很方便的。

无论从流程图哪个位置，如果硬件$\overline{\text{RESET}}$输入端降到地电平（GND）时，都要跳越转移到$\overline{\text{RESET}}$程序开始之处。

$\overline{\text{DMA/BREQ}}$时序只在6809中有效，在6809E型的处理器中$\overline{\text{DMA/BREQ}}$不存在。同样BA、BS状态中的$\overline{\text{HALT}}$/BUS Grant（暂停/总线回答），在6809E型中只有$\overline{\text{HALT}}$。

现在假定发生了快速中断，由流程图可知其过程如下（快速中断标志位（F）要为0）：

MPU被总清后，经过$\overline{\text{RESET}}$时序，从Ⓐ的起点开始到MPU写到SP否？（MPU Write To SP）的所有判断循环之中进行循环。这里，首先判$\overline{\text{HALT}}$，若没有且因$\overline{\text{FIRQ}}$存在，则把该中断锁存起来。接着BA、BS同时写为0。可能输入$\overline{\text{NMI}}$（堆栈指示器S在初始即被给定之时），所以要看一下是否有$\overline{\text{NMI}}$。如果目前认为没有产生$\overline{\text{NMI}}$，下面是$\overline{\text{FIRQ}}\cdot\overline{\text{F}}$当$\overline{\text{FIRQ}}$有效时，而且$\overline{\text{FIRQ}}$标志位也是0时，则$\overline{\text{FIRQ}}\cdot\overline{\text{F}}$的线上有输出并在E标志位写入0。此后把程序计数器和条件码寄存器保留在堆栈区之中。然后在F、I标志位中写入1，接着在总线状态BA写0，在BS中写1。由于条件码寄存器F、I位被置1，即F=1、I=1，所以就不能再接收下面的$\overline{\text{FIRQ}}$和$\overline{\text{IRQ}}$。接着在地址总线上输出向量地址FFF 6，FFF 6地址中的内容送到程序计数器的高位字节，FFF 7地址中的内容送到程序计数器的低位字节之中。程序计数器中送入FFF 6～FFF 7地址中的内容完成时，总线状态BS恢复为0。因此，流程的顺

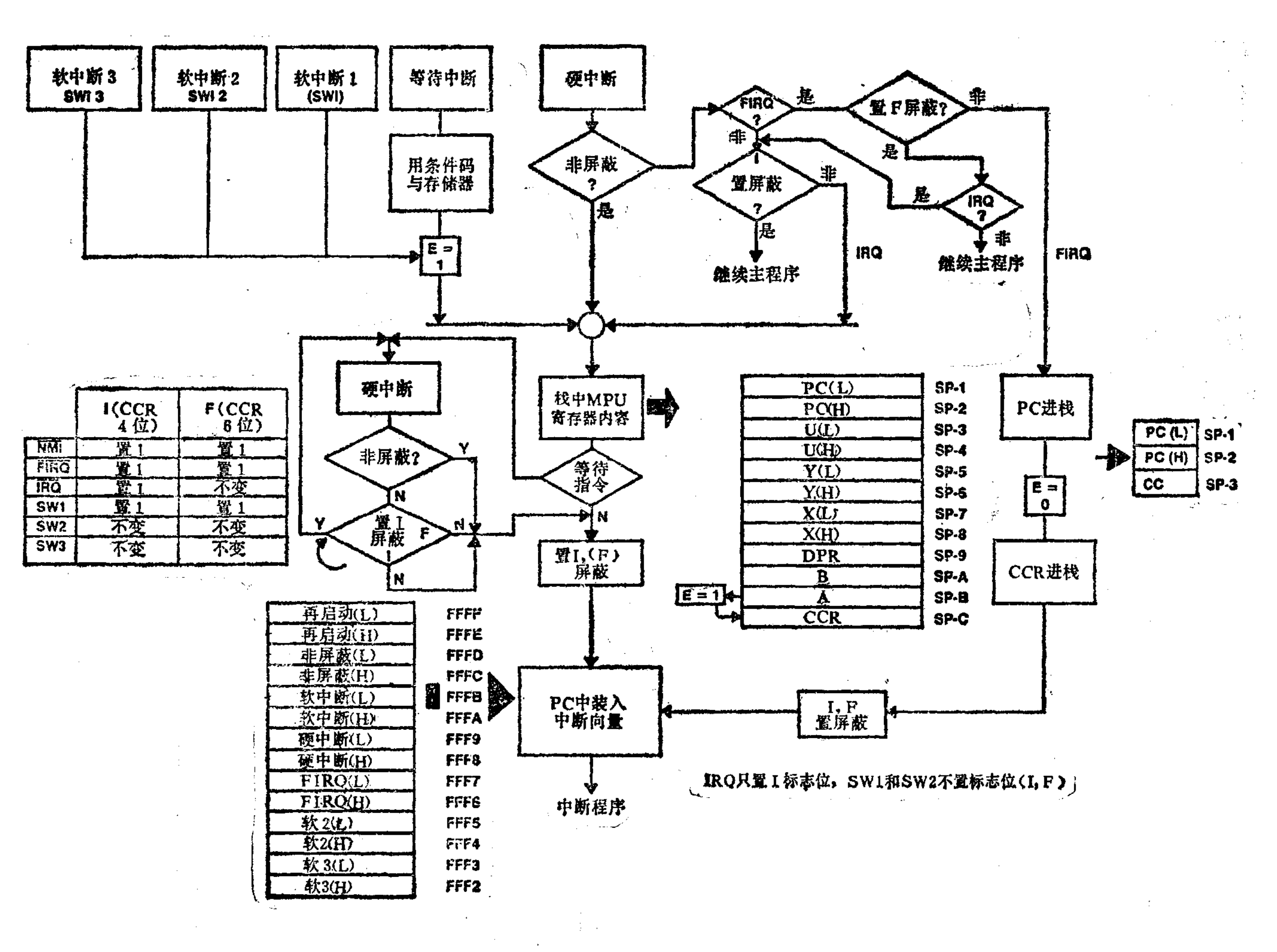
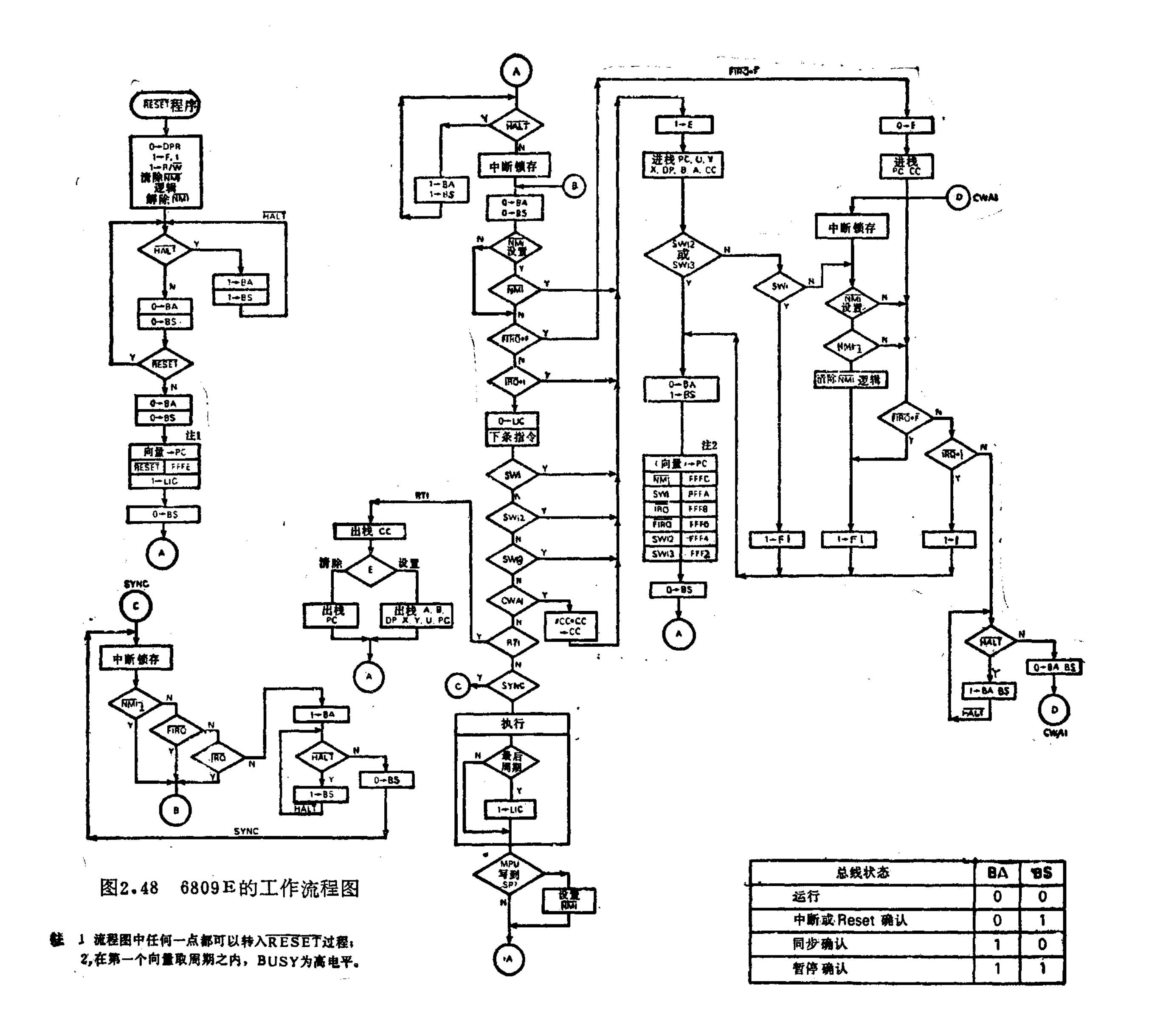


图2.47　6809中断流程图



总线状态	BA	BS
运行	0	0
中断或Reset 确认	0	1
同步确认	1	0
暂停确认	1	1

图2.48　6809E的工作流程图

注　1 流程图中任何一点都可以转入$\overline{\text{RESET}}$过程；
2,在第一个向量取周期之内，BUSY为高电平。

序又转移到Ⓐ。

从Ⓐ处再开始，由于这时带有F＝1，I＝1的条件，$\overline{NMI}$发生并没有受限制，在下条指令（NEXT INST）处取出下条操作码。若没有碰到SWI、SWI 2、SWI 3、RTI或SYNC等指令，则开始执行已经取出的指令。这时，如果要往堆栈指示器S中写入数据（执行LDS指令），则可以接受所设置的硬件中断$\overline{NMI}$。

处理器在输出BS＝0、BA＝0时，则流程图从Ⓐ开始到Ⓐ进行循环，在中途如果遇有RTI指令，则从堆栈中首先恢复条件码，检查相应于条件码中的E标志位，如果E＝0，则只是程序计数器从堆栈中恢复；如果E＝1，则从堆栈恢复全部寄存器内容。

关于6809中断流程的进一步图解说明如图2.47所示。

6809E的工作流程与6809有两点不同，第一，没有$\overline{DMA/BREQ}$的执行过程；第二，在执行指令过程中要判断是否是最后指令周期，是指令最后周期时，使LIC状态线为高电平。6809E的工作流程图如图2.48所示。

第三章　6809的软件

3.1　6809/6809E的指令系统

6809微处理器是硬件和软件设计人员做了极大努力而研究出来的，他们的第一个目标就是要改造6800的设计，做出一种比其它的处理器，特别是在事务和文字处理应用方面性能更为优越的微处理器。处理器的能力的强弱显然是表现在指令系统上，但寻址能力的增强使6809成为了一种极不平常的微处理器。在本章将要集中来讨论这些问题。

6809继承了6800的许多优点，最重要的是6809的设计性能同6800高度地兼容（对语言，只在汇编语言上兼容）。这样就可以使设计人员重新分配6809的操作码，使机器可以更有效地执行。

在6800基础上增加和改进的性能有：

增强寻址方式；

简化定时和控制信号；

给用户设置第二个堆栈；

设置第二个变址寄存器；

在全部储器空间上程序可相对寻址；

支持真正的位置独立程序设计。

任何一种计算机要增强基本指令系统都需靠强化寻址方式。6809有一套很强的寻址方式。它设有59条基本指令，具有1464种不同的指令和寻址方式。所有这些性能都支持现代程序设计方法，其寻址方式包括：

固有寻址	变址寻址	相对寻址
立即寻址	零偏移	短/长相对
直接寻址	常数偏移	程序计数器相对
扩充寻址	累加器偏移	
扩充间接寻址	变址直接	
寄存器寻址		

3.1.1　符号术语定义

1.　存储器寻址的表示法

（　　）＝根据括弧内所写的16位地址决定的存储器内容（8位数据）

EA　　＝有效地址，由寻址方式所决定的存储器地址

M　　＝（EA）＝由有效地址所指定的存储器的内容

MI　　＝存储器立即寻址，接在操作码最后字节的数据

dd　　＝8位的偏移值（可用8位给出，向标号的相对距离）

DDDD　＝16位的偏移值（可用16位给出，向标号的相对距离）

P　　=表示立即、直接、变址、扩充寻址

Q　　=表示累加器、直接、变址、扩充寻址

YYYY　=从－32K字节到＋32K字节的偏移值

ZZ　　=IX、IY、SP、US等指示寄存器

XX　　= 8 位的16进制数

*　　=当前在执行的指令操作码所在的地址

*′　　=下条指令操作码所在的地址

IN　　=只表示变址寻址的符号

#　　=在立即寻址的操作数之前所带的符号

$　　=在16进制数前面所带的符号

%　　=在 2 进制数前面所带的符号

<　　=变址寻址方式时，表示 1 字节的偏移值（参考上述内容）

　　=放在绝对地址之前，表示直接寻址

>　　=变址寻址方式时，表示 2 字节的偏移值

　　=放在绝对地址之前时，表示扩充寻址

,　　表示变址寻址

〔〕　表示间接寻址

2. 运算符号的表示法

←　　=把右边内容代入左边

∩（∧）=每位的逻辑乘（AND）①

∪（∨）=每位的逻辑加（OR）②

⊕　　=每位的异或（半加和）（EXCLUSIVE-OR）

－　　=（上线）=每位的非（NOT）

:　　=连接符

3. 寄存器符号的表示法

ACCA　=A=累加器A

ACCB　=B=累加器B

ACCX　=累加器A或B

ACCA:ACCB=D=双累加器

IX　　=X=变址寄存器X

IY　　=Y=变址寄存器Y

SP　　=S=系统堆栈指示器

US　　=U=用户堆栈指示器

DPR　=DP=直接页面寄存器

CCR　=CC=条件码寄存器

PC　　=PCR=程序计数器。执行指令开始时，程序计数器表示操作码（包括操作数）的

---

① 在6809汇编程序中，也用·表示“与”操作

② 在6809汇编程序中，也用＋表示“或”操作

最后字节的下个地址

R　　= 表示进行运算之前的寄存器，A、B、D、X、Y、U、S、PC、DP、CC（即使写为R，实际上也只是表示为寄存器寻址方式的寄存器可以使用）

R′　　= 进行运算后的寄存器

ALL　= 所有寄存器，即A、B、D、X、Y、S、PC、DP、CC

ZZ　　= 指示寄存器，即X、Y、U、S

MSB　= 最高位

MS BYTE = 高位字节

LS BYTE = 低位字节

IXH　= 变址寄存器X的高位字节

IXL　= 变址寄存器X的低位字节

3.1.2 指令系统

• **ABX** 把ACCB加到IX (add AccB into IX)

记忆符：ABX

动作： IX′←IX + ACCB (IX的内容和ACCB的内容相加后放入IX之中)

条件码： 全无变化

说明：把ACCB的内容看作无符号的二进制数，加在IX的内容上，其结果放入IX之中。

寻找方式：固有

• **ADC** 把存储器内容和进位一起加到寄存器中 (ADd with Carry)

记忆符：ADCA P；ADCB P

动作：R′←R + M + C (把存储器中的数值和C标志位的数值加到寄存器中)

条件码：

H：根据运算，第 3 位产生进位时，被置 1

N：运算后，R中的第 7 位为 1 时，被置 1

Z：运算后，R中的所有各位为 0 时，被置 1

V：根据运算，8 位的 2 的补数产生溢出时，被置 1

C：根据运算，第 7 位产生进位时，被置 1

说明：把存储器的内容和C标志位加到ACCX之中，其结果放入ACCX之中。

寻址方式：存储器：立即　　直接　　变址　　扩充　　间接扩充　　间接变址

寄存器：累加器

• **ADD** 把存储器内容加到寄存器中（8 位）

记忆符：ADDA P；ADDB P

动作：R′←R + M (把存储器的内容加到寄存器中)

条件码：

H：根据运算，第 3 位产生进位时，被置 1

N：运算后，R中第 7 位为 1 时，被置 1

Z：运算后，R中各位全为 0 时，被置 1

V：根据运算，8 位的 2 的补数产生溢出时，被置 1

C：根据运算，第 7 位产生进位时，被置 1

说明：把存储器内容加到ACCX的内容上，结果放入ACCX之中。

寻址方式：存储器：立即　直接　扩充　间接扩充　间接变址

　　　　　寄存器：累加器

• **ADD**　把存储器内容加到寄存器中（16位）

记忆符：ADDD P

动作：R′←R＋<M:M＋1>（把 2 字节的存储器内容加到寄存器内容上）

条件码：

H：无变化

N：运算后，R中的第15位为 1 时，被置 1

Z：运算后，R中所有各种为 0 时，被置 1

V：根据运算，16位的 2 的补数产生溢出时，被置 1

C：在高位字节运算中，第 7 位产生进位时，被置 1

说明：把16位的存储器中数值加到ACCD上，结果放入ACCD之中。

寻址方式：存储器：立即　直接　扩充　间接扩充　间接变址

　　　　　寄存器：双累加器

• **AND**　寄存器和存储器内容进行逻辑乘

记忆符：ANDA P；ANDB P

动作：R′←R∩M（在寄存器和存储器内容之间取AND，并把其结果放入寄存器之中）

条件码：

H：无变化

N：运算后，R中第 7 位为 1 时，被置 1

Z：运算后，R中所有各位都为 0 时，被置 1

V：被置 0

C：无变化

说明：在ACCX内容和存储器内容之间每一位取逻辑乘，其结果放入ACCX之中。

寻址方式：存储器：立即　直接　变址　扩充　间接扩充　间接变址

　　　　　寄存器：累加器

• **AND**　条件码寄存器和数据之间逻辑乘

记忆符：ANDCC＃XX

动作：R′←R∩MI（寄存器内容和操作数数值之间取逻辑乘，其结果放入寄存器之中）

条件码：

CCR′←CCR∩MI（根据MI的数值，可分别被置 0）

说明：在CCR内容和操作数的数值之间，取每一位的逻辑乘，其结果放入CCR之中。该指令可作为各标志位的置 0 指令来使用。

寻址方式：存储器：立即

　　　　　寄存器：CCR

• **ASL**　算术左移（Arithmetic Shift Left）

记忆符：ASL Q

动作：C ← b7 b6 b5 b4 b3 b2 b1 b0 ← O

$C' \leftarrow b_7$, $b_7' \cdots b_1' \leftarrow b_6 \cdots b_0$, $b_0' \leftarrow O$

条件码：

H：不定

N：运算的结果，第7位为1时，被置1

Z：运算的结果，所有各位为0时，被置1

V：运算前操作数（$b_7 \oplus b_6$）的结果进入

C：运算前操作数的第7位内容进入

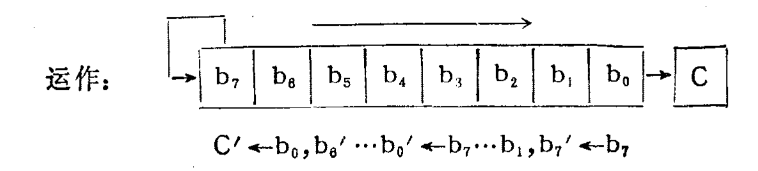
说明：把操作数中所有各位左移一位，0位进入0，第7位的数值进入C标志位。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

• **ASR**　算术右移（Arithmetic Shift Right）

记忆符：ASR　Q

运作：



$C' \leftarrow b_0, b_6' \cdots b_0' \leftarrow b_7 \cdots b_1, b_7' \leftarrow b_7$

条件码：

H：不定

N：运算结果中，第7位为1时，被置1

Z：运算结果中，所有位为0时，被置1

V：无变化

C：运算前操作数的第0位进入

说明：把操作数中所有各位右移一位。第7位不变，第0位的数值进入C标志位。在6800/01/02/03/08处理器中，V标志位要变化，而6809中不变。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

• **BCC**　进位位为0时，分支转移（Branch on Carry Clear）

记忆符：BCC dd；LBCC DDDD

动作：TEMP←MI

C＝0时，PC′←PC＋TEMP（C标志位为0时，则在PC中加上操作数中的偏移值）

条件码：全无变化

说明：检查C标志位数值，若C标志位被置0，则分支转移到有效地址；C标志位若被置1时，则无操作。

寻址方式：存储器：立即

有效寻址：相对　长相对

注释：在无符号的二进制数的减法指令或比较指令之后，如用该指令时，作为确定寄存器

内容比存储器操作数的数值还大或者相等时所用的分支转移指令。

BCS 进位位为1时，分支转移（Branch on Carry Set）

记忆符：BCS dd；LBCS DDDD

动作：TEMP←MI

C＝1时，PC′←PC＋TEMP（C标志位为1时，把操作数的偏移值加在PC上）

条件码：全无变化

说明：检查C标志位之值，如果C标志位被置1，则分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在无符号的二进制数的减法指令或比较指令之后，如用该指令时，作为确定寄存器内容比存储器操作数的数值还小时所用的分支转移指令。

BEQ 相等时，分支转移（Branch on EQual）

记忆符：BEQ dd；LBEQ DDDD

动作：TEMP←MI

Z＝1时，PC′←PC＋TEMP（Z标志位为1时，把操作数的偏移值加在PC上）

条件码：全无变化

说明：检查Z标志位之值，如果Z标志位被置1，则分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：如在减法指令或比较指令之后用此指令，比较的数值（无论是2的补数、还是无符号的数）完全相同时所用的分支转移指令。

BGE 大于或等于零时，分支转移（Branch on Greater or Equal to zero）

记忆符：BGE dd；LBGE DDDD

动作：TEMP←MI

〔N⊕V〕＝0时，PC′←PC＋TEMP（取N标志位和V标志位的异或（半加和），结果为0时，把操作数的偏移值加到PC上。

条件码：全无变化

说明：N标志位和V标志位两者都被置1，或者两者都被置0时的分支转移（即2的补数运算正确执行，其结果为零或者符号为正时的分支转移）。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在2的补数的减法指令或比较指令之后用此指令，作为确定寄存器内容大于或等于存储器操作数的数值时所用的分支转移指令。

BGT 大于时，分支转移（Branch on Greater Than）

记忆符：BGT dd；LBGT DDDD

动作：TEMP←MI

Z∪〔N⊕V〕＝0时，PC′←PC＋TEMP（取N标志位和V标志位的异或，其结果再和Z标志位逻辑加，最后结果为0时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：N标志位和V标志位相等，而且Z标志位又被置0时，分支转移到有效地址。即2的补数运算正确执行，结果为大于零的正数时所用的分支转移指令。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在2的补数减法指令或比较指令之后用此指令，作为确定寄存器的内容大于存储器操作数的数值时所用的分支转移指令。

• **BHI**　大于时，分支转偏 (Branch on HIgher)

记忆符：BHI dd；LBHI DDDD

动作：TEMP←MI

〔CVZ〕＝0时，PC′←PC＋TEMP（取C标志位和Z标志位的逻辑加，其结果为0时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：若C标志位和Z标志位都被置0时，分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在无符号的二进制数的减法指令或比较指令之后用该指令，作为确定寄存器内容大于存储器操作数的数值时所用的分支转移指令。然而一般情况下，在INC/DEC指令、LD/SL指令、TST/CLR/COM指令之后，用该指令是不适合的。

• **BHS**　大于或等于时，分支转移 (Branch on Higher or Same)

记忆符：BHS dd；LBHS DDDD

动作：TEMP←MI

C＝0时，PC′←PC＋TEMP（C标志位为0时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：C标志位被置0时，分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在无符号的二进制数的减法指令或比较指令之后用该指令，作为确定寄存器内容大于或等于存储器操作数的数值时所用的分支转移指令。该指令和BCC指令的操作码和动作相同。然而一般情况下，在INC/DEC指令、LD/ST指令、TST/CLR/COM指令之后，用该指令是不适合的。

• **BIT**　位测试 (BIt Test)

记忆符：BITA；BITB；BIT P

动作：TEMP←R∩M（取R和M间的逻辑乘）

条件码：

H：无变化

N：运算的结果，第7位为1时，被置1

Z：运算的结果，所有各位为0时，被置1

V：被置0

C：无变化

说明：取ACCX的内容和存储器中数值的逻辑乘（AND），根据其结果使条件码改变。
ACCX的内容和存储器中的数值不变。

寻址方式：存储器：立即　直按　变址　扩充　间接扩充　间接变址

寄存器：累加器

• **BLE**　小于或等于0时，分支转移（Branch on Less than or Equal zero）

记忆符：BLE dd；LBLE DDDD

动作：TEMP←MI

Z∪〔N⊕V〕＝1时，PC′←PC＋TEMP（取N标志位和V标志位的异或（半加和），其结果和Z标志位相逻辑加，最后结果为1时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：N标志位和V标志位的数值相异，或者Z标志位被置1时，分支转移到有效地址。
即2的补数运算正确执行，结果为零或符号为负时，分支转移。

寻址方式：存储器：立即

有效寻址：相对　长相对

注释：在2的补数的减法指令或比较指令之后用该指令，作为确定寄存器的内容小于或等于存储器操作数的数值时所用的分支转移指令。

• **BLO**　小于时，分支转移（Branch on LOwer）

记忆符：BLO dd；LBLO DDDD

动作：TEMP←MI

C＝1时，PC′←PC＋TEMP（C标志位为1时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：C标志位为1时，分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　长相对

注释：在无符号的二进制数的减法指令或比较指令之后使用该指令，作为确定寄存器内容比存储器操作数的数值为小时所用的分支转移指令。该指令和BCS指令的操作码和动作是相同的。然而一般情况下，在INC/DEC指令、LD/ST指令、TST/CLR/COM指令之后，用该指令是不适合的。

• **BLS**　小于等于时，分支转移（Branch on Lower or Same）

记忆符：BLS dd；LBLS DDDD

动作：TEMP←MI

(C∪Z)＝1时，PC′←PC＋TEMP（取C标志位和Z标志位的逻辑加，结果为1时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：C标志位或Z标志位被置1时，分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　长相对

注释：在无符号的二进制数的减法指令或比较指令之后使用该指令，作为确定寄存器的内

容小于或等于存储器操作数值时所用的分支转移指令。然而一般情况，在INC/DEC指令、LD/ST指令、TST/CLR/COM指令之后，用该指令是不适合的。

• **BLT** 小于0时，分支转移 (Branch on Less Than zero)

记忆符：BLT dd；LBLT DDDD

动作：TEMP←MI

〔N⊕V〕＝1时，PC′←PC＋TEMP（取N标志位和V标志位之异或（半加和）结果为1时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：N标志位和V标志位的数值相等时，分支转移到有效地址。即2的补数运算正确执行结果的符号为负时所用的分支转移指令。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在2的补数的减法指令或比较指令之后用该指令，作为确定寄存器的内容小于存储器操作数的数值时所用的分支转移指令。

• **BMI** 为负时，分支转移 (Branch on MInus)

记忆符：BMI dd；LBMI DDDD

动作：TEMP←MI

N＝1时，PC′←PC＋TEMP（N标志位为1时，把操作数的偏移值加到PC上）

条件码：全无变化

说明：N标志位被置1时，分支转移到有效地址。即2的补数运算结果符号位为负时所用的分支转移指令

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在2的补数运算之后用该指令，结果为负时所用的分支转移指令（但是，如果没有检查V标志位时，则不知道结果是否为正确的2的补数）。

• **BNE** 不相等时，分支转移 (Branch on Not Equal)

记忆符：BNE dd；LBNE DDDD

动作：TEMP←MI

Z＝0时，PC′←PC＋TEMP（Z标志位被置0时，操作数的偏移值加到PC上）

条件码：全无变化

说明：Z标志位被置0时，分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释：在二进制数的减法指令或比较指令之后用该指令，作为确定寄存器内容和存储器操作数的数值不相等时所用的分支转移指令。

• **BPL** 为正时，分别转移 (Branch on PLus)

记忆符：BPL dd；LBPL DDDD

动作：TEMP←MI

N＝0时，PC′←PC＋TEMP（N标志位被置0时，把操作数偏移值加到PC上）

条件码：全无变化

说明：N标志位被置0时，分支转移到有效地址。即2的补数运算结果符号为正时，分支转移。

寻址方式：存储器：立即

有效寻址：相对　　长相对

注释： 在2的补数运算之后用该指令，判断结果为正时所用的分支转移指令（但是，如果没有检查V标志位时，则不知道结果是否为正确的2的补数）。

• **BRA**　无条件分支转移（BRanch Always）

记忆符：BRA dd；LBRA DDDD

动作：TEMP←MI

PC′←PC+TEMP（把操作数的偏移值加到PC上）

条件码：全无变化

说明：与各标志位之值无关，向有效地址分支转移。

寻址方式：存储器：立即

有效寻址：相对　　长相对

• **BRN**　非分支转移（BRanch Never）

记忆符：BRN dd；LBRN DDDD

动作：TEMP←MI（无执行动作）

条件码：全无变化

说明：是不进行转移的分支转移指令，该指令即为NOP指令，但具有伪操作数。因此，可以认为BRN是具有1字节的空操作指令，LBRN是2字节的空操作指令。

寻址方式：存储器：立即

有效寻址：相对　　长相对

• **BSR**　向子程序分支转移（Branch to SubRoutine）

记忆符：BSR dd；LBSR DDDD

动作：TEMP←MI

SP′←SP-1，(SP)←PCL

SP′←SP-1，(SP)←PCH

PC′←PC+TEMP(PC保留到系统堆栈之中，把操作数的偏移值加到PC上)

条件码：全无变化

说明：把程序计数器保留到系统堆栈之中，分支转移到有效地址。

寻址方式：存储器：立即

有效寻址：相对　　长相对

• **BVC**　溢出标志位为0时，分支转移（Branch on oVerflow Clear）

记忆符：BVC dd；LBVC DDDD

动作：TEMP←MI

V=0时，PC′←PC+TEMP(V标志位为0时，把操作数的偏移值加到PC上)

条件码：全无变化

说明：V标志位被置0时，分支转移到有效地址。即2的补数运算结果为正确的补数时，

分支转移。

寻址条件：存储器：立即

有效寻址：相对　　长相对

注释：在2的补数运算之后用该指令，判断没有产生溢出时所用的分支转移指令。

· **BVS**　溢出标志位为1时，分支转移 (Branch on oVerflow Set)

记忆符：BVS dd；LBVS DDDD

动作：TEMP←MI

V＝1时，PC′←PC＋TEMP (V位标志位为1时，把操作数的偏移值加到PC上)

条件码：全无变化

说明：V标志位被置1时，分支转移到有效地址。即2的补数运算结果不正确时，分支转移。

寻址方式：存储器：立即

有效地址：相对　　长相对

注释：在2的补数运算之后用该指令，判断有溢出存在时所用的分支转移指令。另外，该指令还可在检查二进制浮点规格化的ASL或LSL指令之后使用。

· **CLR**　清零 (CLeaR)

记忆符：CLR Q

动作：TEMP←M

M←$00 (把$00写入M中)

条件码：

H：不变

N：被置0

Z：被置1

V：被置0

C：被置0

说明：ACCX或存储器的所有各位为0。为保持和6800处理器之间的兼容性，C标志位被置0。

寻址方式：累加器　　直接　　变址　　扩充　　间接扩充　　间接变址

· **CMP**　寄存器和存储器之间比较（8位）(CoMPare)

记忆符：CMPA P；CMPB P

动作：TEMP←R－M〔即TEMP←R＋$\overline{M}$＋1〕(从R的内容减去存储器的数值)

条件码：

H：不定

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：根据运算，第8位的2之补数产生溢出时，被置1

C：根据运算，第7位没有产生进位时，被置1

说明：累加器内容和存储器数值进行比较，根据结果使条件码改变。存储器或累加器的内容不变。C标志位代表借位，累加器内容比存储器的数值小时，该位被置1。

寻址方式：存储器：立即　直接　变址　扩充　间接扩充　间接变址

寄存器：累加器

标志位的意义：

如 $(N\oplus V)=1$，$R<M$（2的补数）

如 $C=1$，$R<M$（无符号二进制数）

如 $Z=1$，$R=M$

· CMP 寄存器和存储器之间比较（16位）（CoMPare）

记忆符：CMPD P；CMPX P；CMPY P；CMPU P；CMPS P

动作：$TEMP\leftarrow R-<M:M+1>$〔即，$TEMP\leftarrow R+<\overline{M:M+1}>+1$〕（从R的内容减去2字节的存储器的数值）

条件码：

H：不变

N：运算结果，第15位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：根据运算，第16位的2之补数产生溢出时，被置1

C：根据运算，从第15位没有产生进位时，被置1

说明：16位寄存器内容和16位的存储器的数值进行比较，根据其结果改变条件码。寄存器和存储器中的内容不变。C标志位表示借位，当寄存器内容比存储器中数值小时，该位被置1

寻址方式：存储器：立即　直接　变址　扩充　间接扩充　间接变址

寄存器：双累加器　指示寄存器（X、Y、S或U）

标志位的意义：

如 $(N\oplus V)=1$，$R<M$（2的补数）

如 $C=1$，$R<M$（无符号二进制数）

如 $Z=1$，$R=M$

· COM 求1的补数（COMplement）

记忆符：COM Q

动作：$M'\leftarrow 0+\overline{M}$（把M的各位全变为反码）

条件码：

H：不变

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：被置0

C：被置1

说明：取累加器或存储器中内容为1的补数（亦称逻辑反码）。为使C位和6800处理器兼容而被置1

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

注释：在无符号的二进制数使用该指令时，除BEQ和BNE指令外，也许不会正常工作。如对2的补数进行运算，可以使用所有带符号的分支转移指令(BLT/BLE/BGE/BGT

/BEQ/BNE)。

• **CWAI** 清零条件码并等待中断 (**Clear and WAit for Interrupt**)

记忆符： CWAI # $ xx

E	F	H	I	N	Z	V	C

动作： CCR′←CCR∩MI (置0中断屏蔽位)

E位置1 (表示所有内部寄存器都保留)

SP′←SP－1,(SP)←PCL

SP′←SP－1,(SP)←PCH

SP′←SP－1,(SP)←USL

SP′←SP－1,(SP)←USH

SP′←SP－1,(SP)←IYL

SP′←SP－1,(SP)←IYH

SP′←SP－1,(SP)←IXL

SP′←SP－1,(SP)←IXH

SP′←SP－1,(SP)←DPR

SP′←SP－1,(SP)←ACCB

SP′←SP－1,(SP)←ACCA

SP′←SP－1,(SP)←CCR

MI的数值

$ FF＝IRQ位、FIRQ位都不变

$ EF＝允许IRQ

$ BF＝允许FIRQ

$ AF＝允许IRQ和FIRQ

(取CCR和MI之逻辑乘，其结果在CCR之中。其次，E标志位置1时，除系统堆栈指示器外，所有内部寄存器都保留在堆栈之中)

条件码：可按照操作数的数值被置0。E标志位被置1。

说明：取一字节操作数数值和条件码寄存器之逻辑乘，其结果存入条件码寄存器之中。按照这种方法可使中断屏蔽位被置0。其次，除系统堆栈指示器外全部内部寄存器都被保留在系统堆栈之内，而进入等待中断的状态。中断发生时，不再进行寄存器的的保留。该指令和6800处理器的CLI指令＋WAI指令具有同样的动作，总线不处于高阻抗状态。

寻址方式：存储器：立即

注释：FIRQ中断如果发生在该指令之后，因为内部寄存器状态全被保留，可以很快转移到中断处理程序。对RTI指令，在检查了被保留的CCR的E标志位之后，自动地恢复内部寄存器内容。

• **DAA** ACCA的十进制加法调整 (Decimal Addition Adjust)

记忆符：DAA

动作：ACCA′←ACCA＋CF(MSN):CF(LSN) (把8位的调整因子加到ACCA之中)

其中CF是4位的调整因子，其内容如下：对每个半字节（表示BCD数4位）CF可分别确定为6还是0。

对低位半字节（LSN）：

CF（LSN）=0时，为1）H=1

或2）LSN>9

对高位半字节（MSN）：

CF（MSN）=6时，为1）C=1

或2）MSN>9

或3）MSN>8但LSN>9

条件码：

H：不变

N：运算结果，累加器的第7位为1时，被置1

Z：运算结果，累加器所有各位为0时，被置1

V：不定

C：根据运算；从第7位产生进位，或者在运算之前C标志位被置1时，被置1

说明：如果在ACCA中的一字节加法指令（ADDA或ADCA）之后，使用DAA指令时，即可得到考虑C标志位的二-十进制的加法（BCD加法）。这时，加数和被加数必须是二-十进制数（即每个半字节必须是0到9的数值）。当进行二字节以上的多精度加法时，必须把原来DAA指令发生的进位在下一位进行加法时相加在一起。而且在这以后还必须再做一次DAA指令。

寻址方式：固有（ACCA）

- **DEC** 减1（DECrement）

记忆符：DEC Q

动作：M′←M－1［即M′←M＋$FF］，（从M中减1）

条件码：

H：不变

N：运算结果，操作数的第7位为1时，被置1

Z：运算结果，操作数所有各位为0时，被置1

V：运算前的操作数为二进制10000000时，被置1

C：不变

说明：从操作数所指定的存储器或寄存器中的内容减1，C标志位不变。因此，在多倍精度计算中，作为循环的计数器可以使用DEC指令进行。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

注释：对无符号的二进制数使用该指令时，可以与条件判断指令BEQ或BNE正常工作时相配合使用。对2的补数使用该指令时，可与所有带符号的分支转移指令（BLT/BLE/BGE/BGT/BEQ/BNE）相配合使用。

- **EOR** 异或（半加和）（Exclusive OR）

记忆符：EORA P；EORB P

动作：R′←R⊕M（取R和M间的异或，结果存入R中）

条件码：

H：不变

N：运算结果，寄存器的第7位为1时，被置1

Z：运算结果，寄存器所有各位为0时，被置1

V：被置0

C：不变

说明：取寄存器内容和存储器中数值的异或，结果放入寄存器之中。

寻址方式：存储器：直接　扩充　立即　变址　间接扩充　间接变址

寄存器：累加器

• **EXG** 寄存器交换（EXchanGe registers）

记忆符：EXG R 1，R 2

动作：R 1 ←→ R 2（交换R 1和R 2中的内容）

条件码：交换寄存器一方不用条件码寄存器的不变

说明：用后缀字节0～3位指定一方寄存器，4～7位指定另一方寄存器。其规定如下：

0000 = A:B = D	1000 = A
0001 = X	1001 = B
0010 = Y	1010 = CCR
0011 = US	1011 = DPR
0100 = SP	1100 = 未定义
0101 = PC	1101 = 未定义
0110 = 未定义	1110 = 未定义
0111 = 未定义	1111 = 未定义

寄存器之间交换时，各寄存器的位数应相等才能彼此交换，如8位同8位、16位同16位。

寻址方式：寄存器（固有）

• **INC** 加1（INCrement）

记忆符：INC Q

动作：M′←M＋1（在M中加1）

条件码：

H：不变

N：运算结果，操作数的第7位为1时，被置1

Z：运算结果，操作数的所有各位为0时，被置1

V：运算前的操作数的内容为二进制数01111111时，被置1

C：不变

说明：对操作数所指定的寄存器或存储器中的内容加1，因为C标志位的内容不变，所以INC指令可以在多倍精度计算时作为循环计数器来使用。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

注释：对无符号的二进制数使用该指令时，可以与条件判断指令BEQ或BNE正常工作时相配合使用。对2的补数使用该指令时，可与所有带符号的分支转移指令（BLT/BLE/BGE/BGT/BEQ/BNE）相配合使用。

• **JMP** 跳跃转移到有效地址 (JuMP to effective address)

记忆符：JMP

动作：PC′←EA（把有效地址数值放入PC之中）

条件码：全无变化

说明：按操作数所给的有效地址跳越转移。

寻址方式：直接　　变址　　扩充　　间接扩充　　间接变址

• **JSR** 跳越转移到子程序的有效地址上 (JoMP to subroutine at effective address)

记忆符：JSR

动作：SP′←SP－1，　(SP)←PCL

　　　SP′←SP－1，　(SP)←PCH

　　　PC′←EA（PC内容保留在堆栈中，有效地址放入PC之中）

条件码：全无变化

说明：返回地址被保留在系统堆栈之中，跳越转移到操作数所给的有效地址上。

寻址方式：直接　　变址　　扩充　　间接扩充　　间接变址

• **LD** 把存储器内容装入寄存器（8位） (LoaD register from memory——8 Bit)

记忆符：LDA P；LDB P

动作：R′←M（把存储器数值加入寄存器）

条件码：

　H：不变

　N：被装入数据的第7位为1时，被置1

　Z：被装入数据所有各位为0时，被置1

　V：被置0

　C：不变

说明：把按操作数所给的存储器内容加入寄存器之中。

寻址方式：存储器：立即　　直接　　变址　　扩充　　间接扩充　　间接变址

　　　　　寄存器：累加器

• **LD** 把存储器内容装入寄存器（16位） (LoaD register from memory——16Bit)

记忆符：LDD P；LDX P；LDY P；LDS P；LDU P

动作：R′←<M:M＋1>（把2字节的存储器数值加入寄存器）

条件码：

　H：不变

　N：装入数据的第15位为1时，被置1

　Z：装入数据中所有各位为0时，被置1

　V：被置0

　C：不变

说明：把按操作数所给的相邻的2字节的存储器中的数值加入16位的寄存器。

寻址方式：存储器：立即　　直接　　变址　　扩充　　间接扩充　　间接变址

　　　　　寄存器：双累加器　　指示器（X、Y、S、U）

• **LEA** 装入有效地址 (Load Effective Address)

记忆符：LEAX；LEAY；LEAS；LEAU

动作：R′←EA（把有效地址装入寄存器中）

条件码：

H：不变

N：不变

Z：LEAX、LEAY：装入的数据中所有各位为0时，被置1

LEAS、LEAU：不变

V：不变

C：不变

说明：把按照寻址方式所定的有效地址装入指示寄存器。LEAX或LEAY指令因使Z位改变，可作为计数器使用，而且和6800处理器INX/DEX指令具有兼容性。LEAU和LEAS指令因不改变Z位，所以当使Z作为参数返回主程序时，允许清除堆栈，而且允许和6800处理器的INS/DES指令之间具有兼容性。

寻址方式：存储器：变址　　间接扩充　　间接变址

寄存器：指示器（X、Y、S、U）

• **LSL** 逻辑左移（Logical Shift Left）

记忆符：LSL Q

动作：| C | ← | b_7 | b_6 | b_5 | b_4 | b_3 | b_2 | b_1 | b_0 | ← O

$C' \leftarrow b_7, b'_7 \cdots b'_1 \leftarrow b_6 \cdots b_0, b_0' \leftarrow 0$

条件码：

H：不定

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：运算前操作数的（$b_7 \oplus b_6$）的结果放入其中

C：运算前操作数的第7位放入其中

说明：把按操作数所指定的寄存器或存储器的所有各位都左移一位，在第零位中放入0，第7位的数值进入C标志位。该指令与ASL指令的动作和机器码相同。

寻址方式：累加器　　直接　　变址　　扩充　　间接扩充　　间接变址

• **LSR** 逻辑右移（Logical Shift Right）

记忆符：LSR Q

——→

动作：O→ | b_7 | b_6 | b_5 | b_4 | b_3 | b_2 | b_1 | b_0 | →C

$C' \leftarrow b_0,\ b_0' \cdots b_6' \leftarrow b_1 \cdots b_7,\ b_7' \leftarrow 0$

条件码：

H：不变

N：被置0

Z：运算结果，所有各位为0时，被置1

V：不变

C：运算前操作数的第0位被装入

说明：操作数中所有各位右移一位，第7位进入0，C标志位进入第0位之值。6800处理器中，V标志位也要变化。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

• **MUL**　累加器内相乘（MULtiply accumulator）

记忆符：MUL

动作：ACCA′：ACCB′←ACCA×ACCB（ACCA和ACCB进行乘法，结果放在ACCD之中）

条件码：

H：不变

N：不变

Z：运算结果，所有各位为0时，被置1

V：不变

C：运算结果，ACCB的第7位为1时，被置1

说明：使两个累加器中内容按无符号二进制数进行乘法，其结果放入两个累加器之中。在无符号乘法中，可以作多倍精度运算。C标志位在高位字节作舍入近似计算时使用，即如果按MUL，ADCA # 0顺序时，乘法结果的近似值放在ACCA之中。

寻址方式：固有

• **NEG**　2的补数（NEGate）

记忆符：NEG Q

动作：$M' \leftarrow 0 - M$〔即，$M' \leftarrow \overline{M} + 1$〕（变中的各位，全部变反后，加1）

条件码：

H：不定

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：运算前操作数的内容，为二进制数10000000时，被置1

C：根据运算，第7位有进位时，被置1

说明：把操作数内容取2的补数后其数值放入操作数之中。此时C标志位表示借位，进入的是与进位相反的数值。操作数的内容为$80时，该指令执行后也得到相同的数值，这时只有在V标志位置1。还有，在$00时执行该指令之后，其数值也不变，只有在这时C标志位被置0。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

标志位的意义：

$(N \oplus V) = 1$时，表示$0 < M$的情况（2的补数）

$C = 1$时，表示$0 < M$的情况（无符号二进制数）

$Z = 1$时，表示$0 = M$

• **NOP**　空操作（No OPeration）

记忆符：NOP

条件码：全无变化

说明：该指令是一字节指令，只是使程序计数器的数值加 1，寄存器和存储器的内容不变。

寻址方式：固有

- **OR** 寄存器和存储器进行逻辑加（inclusive OR）

记忆符：ORA P；ORB P

动作：R′←R∪M（取R和M之逻辑加，结果放入R中）

条件码：

H：不变

N：运算结果，第 7 位为 1 时，被置 1

Z：运算结果，所有各位为 0 时，被置 1

V：被置 0

C：不变

说明：取ACCX内容和存储器内容的逻辑加，结果放在ACCX之中。

寻址方式：存储器：立即　直接　变址　扩充　间接扩充　间接变址

寄存器：累加器

- **OR** 存储器和CCR进行逻辑加（inclusive OR）

记忆符：ORCC # ××

动作：R′←R ∪ MI（取R和MI的逻辑加，结果放入R中）

条件码：CCR′←CCR ∪ MI

说明：取CCR和操作数的数据逻辑加，结果放在CCR之中。该指令用于进行中断屏蔽（禁止中断），或者其它的标志位需进行置 1 时使用。

寻址方式：存储器：立即

寄存器：CCR

- **PSHS** 向系统堆栈保留寄存器内容（PuSH on System stack）

记忆符：PSHS 寄存器表

PSHS #Label



PC	U	Y	X	DP	B	A	CC

保留程序

动作：MI的第 7 位为 1 时，SP′←SP－1，(SP)←PCL

SP′←SP－1，(SP)←PCH

MI的第 6 位为 1 时，SP′←SP－1，(SP)←USL

SP′←SP－1，(SP)←USH

MI的第 5 位为 1 时，SP′←SP－1，(SP)←IYL

SP′←SP－1，(SP)←IYH

MI的第 4 位为 1 时，SP′←SP－1，(SP)←IXL

SP′←SP－1，(SP)←IXH

MI的第 3 位为 1 时，SP′←SP－1，(SP)←DPR

MI的第 2 位为 1 时，SP′←SP－1，(SP)←ACCB

MI的第 1 位为 1 时，SP′←SP－1，(SP)←ACCA

MI的第 0 位为 1 时，SP′←SP－1，(SP)←CCR

(对应操作数的数值把寄存器保留在系统堆栈之中)

条件码：全无变化

说明：除系统堆栈指示器外，使CPU全部的内部寄存器或部分内部寄存器保留在系统堆栈之中。

寻址方式：存储器：立即

寄存器：寄存器

• **PSHU** 向用户堆栈保留寄存器内容 (PuSH on User stack)

记忆符：PSHU寄存器表

PSHU #Label



PC	S	Y	X	DP	B	A	CC

——→
保留序顺

动作：MI的第 7 位为 1 时，US′←US－1，(US)←PCL

US′←US－1，(US)←PCH

MI的第 6 位为 1 时，US′←US－1，(US)←SPL

US′←US－1，(US)←SPH

MI的第 5 位为 1 时，US′←US－1，(US)←IYL

US′←US－1，(US)←IYH

MI的第 4 位为 1 时，US′←US－1，(US)←IXL

US′←US－1，(US)←IXH

MI的第 3 位为 1 时，US′←US－1，(US)←DPR

MI的第 2 位为 1 时，US′←US－1，(US)←ACCB

MI的第 1 位为 1 时，US′←US－1，(US)←ACCA

MI的第 0 位为 1 时，US′←US－1，(US)←CCR

(对应操作数的数值把寄存器保留在用户堆栈之中)

条件码：全无变化

说明：除用户堆栈指示器外，使CPU全部的内部寄存器，或部分内部寄存器内容保留在用户堆栈之中。

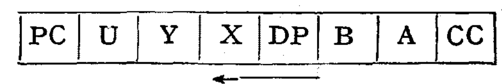
寻址方式：存储器：立即

寄存器：寄存器

• **PULS** 从系统堆栈恢复寄存器 (PULl from System Stack)

记忆符：PULS寄存器表

PULS #Label



PC	U	Y	X	DP	B	A	CC

←——
恢复顺序

动作：MI的第 0 位为 1 时，CCR′←(SP)，SP′←SP＋1
MI的第 1 位为 1 时，ACCA′←(SP)，SP′←SP＋1
MI的第 2 位为 1 时，ACCB′←(SP)，SP′←SP＋1
MI的第 3 位为 1 时，DPR′←(SP)，SP′←SP＋1
MI的第 4 位为 1 时，IXH′←(SP)，SP′←SP＋1
IXL′←(SP)，SP′←SP＋1
MI的第 5 位为 1 时，IYH′←(SP)，SP′←SP＋1
IYL′←(SP)，SP′←SP＋1
MI的第 6 位为 1 时，USH′←(SP)，SP′←SP＋1
USL′←(SP)，SP′←SP＋1
MI的第 7 位为 1 时，PCH′←(SP)，SP′←SP＋1
PCL′←(SP)，SP′←SP＋1
(对应操作数的数值从系统堆栈中恢复寄存器内容)

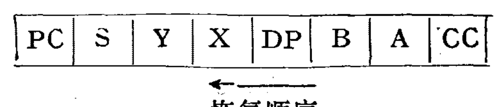
条件码：CCR的内容如从堆栈中被恢复，则为被存在堆栈中的数值，而其它情况不变

说明：除系统堆栈指示器外，CPU的全部内部寄存器或一部分内部寄存器将从系统堆栈被恢复。当只恢复一个寄存器时，可以使用LD指令中的自动加 1 的变址方式（例：LDA，S＋）。但是，此时应注意标志位会发生变化。

寻址方式：存储器：立即
寄存器：寄存器

· PULU 从用户堆栈中恢复寄存器（PULl from User stack）

记忆符：PULU 寄存器表
PULU #Label



动作：MI的第 0 位为 1 时，CCR′←(US)，US′←US＋1
MI的第 1 位为 1 时，ACCA′←(US)，US′←US＋1
MI的第 2 位为 1 时，ACCB′←(US)，US′←US＋1
MI的第 3 位为 1 时，DPR′←(US)，US′←US＋1
MI的第 4 位为 1 时，IXH′←(US)，US′←US＋1
IXL′←(US)，US′←US＋1
MI的第 5 位为 1 时，IYH′←(US)，US′←US＋1
IYL′←(US)，US′←US＋1
MI的第 6 位为 1 时，SPH′←(US)，US′←US＋1
SPL′←(US)，US′←US＋1
MI的第 7 位为 1 时，PCH′←(US)，US′←US＋1
PCL′←(US)，US′←US＋1
(对应的操作数的数值，从用户堆栈中恢复寄存器内容)

条件码：如果CCR的内容从堆栈中恢复，则变为原在堆栈中的数值，其它情况不变。

说明： 除用户堆栈指示器外，CPU全部内部寄存器，或部分内部寄存器的内容，从用户堆栈中被恢复。当只恢复一个寄存器的内容时，可以使用 LD 指令的自动加 1 变址方式（例LDX，U++）。但是，此时应注意标志位会发生变化。

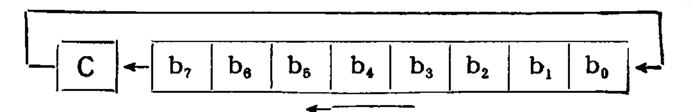
寻址方式：存储器：立即

寄存器：寄存器

• **ROL** 循环左移（ROtate Left）

记忆符：ROL Q

动作：



$C' \leftarrow b_7$， $b'_7 \cdots b_1' \leftarrow b_6 \cdots b_0$， $b_0' \leftarrow C$

（是C位和操作数共 9 位在一起进行循环左移）

条件码：

H：不变

N：运算结果，第 7 位为 1 时，被置 1

Z：运算结果，所有各位为 0 时，被置 1

V：运算前操作数中的（$b_7 \oplus b_6$）的结果进入该位

C：运算前操作数中的第 7 位进入该位

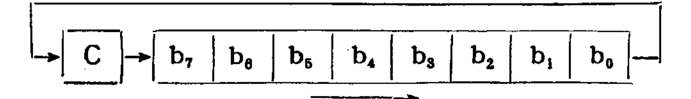
说明： 操作数所有各位和C标志位被循环左移一次。C标志位将进入第 7 位内容， 0 位中将进入C标志位内容。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

• **ROR** 循环右移（ROtate Right）

记忆符：ROR Q

动作：



$C' \leftarrow b_0, b_6' \cdots b_0' \leftarrow b_7 \cdots b_1, b_7' \leftarrow C$（是操作数和C标志位共9位在一起进行循环右移）

条件码：

H：不变

N：运算结果，第 7 位为 1 时，被置 1

Z：运算结果，所有各位为 1 时，被置 1

V：不变

C：运算前操作数中的第 0 位进入该位。

说明： 是把 8 位操作数和C标志位合在一起所有 9 位数循环右移一次。在6800处理器中V标志位也不变。

寻址方式：累加器　直接　变址　扩充　间接扩充　间接变址

• **RTI** 中断返回（ReTurn from Interrupt）

记忆符：RTI

动作：CCR′←(SP)，SP′←SP＋1

CCR中的E为1时：

ACCA′←(SP),SP′←SP＋1

ACCB′←(SP),SP′←SP＋1

DPR′←(SP)，SP′←SP＋1

IXH′←(SP)，SP′←SP＋1

IXL′←(SP)，SP′←SP＋1

IYH′←(SP)，SP′←SP＋1

IYL′←(SP)，SP′←SP＋1

USH′←(SP)，SP′←SP＋1

USL′←(SP)，SP′←SP＋1

PCH′←(SP)，SP′←SP＋1

PCL′←(SP)，SP′←SP＋1

CCR中的E位为0时：

PCH′←(SP)，SP′←SP＋1

PCL′←(SP)，SP′←SP＋1

（首先从堆栈中恢复CCR内容，如果E标志位被置1，则所有内部寄存器的内容被恢复。E标志位如果被置0，则只有PC内容被恢复）

条件码：从堆栈中被恢复

说明：被保留在堆栈中的CPU内部寄存器的内容从系统堆栈中恢复，返回被中断了的程序。如果在恢复了的CCR中的E标志位被置0时，则只是恢复返回地址和CCR。

寻址方式：固有

• **RTS** 从子程序返回（ReTurn from Subrotine）

记忆符：RTS

动作：RCH′←(SP)，SP′←SP＋1

PCL′←(SP)，SP′←SP＋1（从堆栈中恢复PC内容）

条件码：全无变化

说明：从系统堆栈中把返回地址恢复到程序计算器中，从子程序返回。

寻址方式：固有

• **SBC** 带借位的减法（SuBtract with Carry＝borrow）

记忆符：SBCA P；SBCB P

动作：$R' \leftarrow R-M-C$〔即$R' \leftarrow R+\overline{M}+\overline{C}$〕（从寄存器中减去存储器的数值和借位）

条件码：

H：不定

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：根据8位2的补数运算发生溢出时，被置1

C：运算时，第7位无进位时，被置1

说明：从累加器的内容中减去存储器的数值和借位位（用C位表示），其结果放入累加器。C标志位表示借位，累加器的内容如果小时，被置 1

寻址方式：存储器：立即　直接　变址　扩充　间接扩充　间接变址

寄存器：累加器

• **SEX**　2 的补数的扩充（(Sign EXtended)

记忆符：SEX

动作：ACCB的第 7 位为 1 时，ACCA′←$FF

ACCB的第 7 位为 0 时，ACCA′←$00

条件码：

H：不变

N：运算结果，ACCA的第 7 位为 1 时，被置 1

Z：运算结果，ACCD的所有各位为 0 时，被置 1

V：不变

C：不变

说明：该指令把ACCB中 8 位 2 的补数，变换为16位的 2 的补数，放入双累加器之中。

寻址方式：固有

• **ST**　把寄存器内容存在存储器之中（8 位）

(STore register into memory——8 Bits)

记忆符：STA P；STB P

动作：M′←R（把寄存器的内容放入存储器）

条件码：

H：不变

N：被存数据的第 7 位为 1 时，被置 1

Z：被存数据所有各位为 0 时，被置 1

V：被置 0

C：不变

说明：把累加器的内容放入操作数所指定的有效地址的存储器单元之中

寻址方式：存储器：直接　变址　扩充　间接扩充　间接变址

寄存器：累加器

• **ST**　把寄存器的内容存在存储器之中（16位）

(STore register into memory——16 Bits)

记忆符：STD P；STX P；STY P；STS P；STU P

动作：<M′：M+1′>←R（把寄存器的内容存入存储器两个单元中）

条件码：

H：不变

N：被存数据的第15位为 1 时，被置 1

Z：被存数据的所有各位为 0 时，被置 1

V：被置 0

C：不变

说明：把16位的寄存器内容存入由操作数所指定有效地址相邻的2字节 存 储 器 单 元 之中。

寻址方式：存储器：直接　变址　扩充　间接扩充　间接变址

寄存器：双累加器　指示器（X、Y、S、U）

• **SUB** 寄存器内容减去存储器内容（8位）

(SUBtract memory from register—— 8 Bits)

记忆符：SUBA P；SUBB P

动作：$R' \leftarrow R - M$〔即$R' \leftarrow R + \overline{M} + 1$〕（从寄存器的内容减去存储器的数值）

条件码：

H：不定

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：运算时，8位2的补数溢出时，被置1

C：运算时，第7位无进位时，被置1

说明：从累加器内容减去存储器数值，结果放入累加器之中。C标志位表示借位，累加器的内容比存储器中数值小时，被置1。

寻址方式：存储器：　立即　直接　变址　扩充　间接扩充　间接变址

寄存器：累加器

标志位的意义：

$(N \oplus V) = 1$时，表示$R < M$（2的补数）

$C = 1$时，表示$R < M$（无符号二进制数）

$Z = 1$时，表示$R = M$

• **SUB** 寄存器内容减去存储器内容（16位）

(SUBtract memory from register——16 Bits)

记忆符：SUBD P

动作：$R' \leftarrow R - \langle M:M+1 \rangle$〔即$R' \leftarrow R + \langle \overline{M:M+1} \rangle + 1$〕

（寄存器的内容减去存储器二个字节的数值）

条件码：

H：不变

N：运算结果，第15位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：运算时，16位2的补数溢出时，被置1

C：在高位字节运算中，第7位无进位时，被置1

说明：从双累加器的内容中，减去存储器相邻二个字节的数值，其结果放入双累加器中。C标志位表示借位，双累加器的内容比存储器的数值小时，被置1。

寻址方式：存储器：立即　直接　变址　扩充　间接扩充　间接变址

寄存器：双累加器

标志位的意义：

$(N \oplus V) = 1$时，表示$R < M$（2的补数）

C＝1时，表示R<M（无符号的二进制数）

Z＝1时，表示R＝M

• **SWI** 软件中断（SoftWare Interrupt）

记忆符：SWI

动作：把E标志位置1（全部保留寄存器）

SP′←SP－1，（SP）←PCL
SP′←SP－1，（SP）←PCH
SP′←SP－1，（SP）←USL
SP′←SP－1，（SP）←USH
SP′←SP－1，（SP）←IYL
SP′←SP－1，（SP）←IYH
SP′←SP－1，（SP）←IXL
SP′←SP－1，（SP）←IXH
SP′←SP－1，（SP）←DPR
SP′←SP－1，（SP）←ACCB
SP′←SP－1，（SP）←ACCA
SP′←SP－1，（SP）←CCR
置1 I标志位和F标志位（禁止中断）
PC′←（$FFFA）：（$FFFB）
（置位E、I、F标志位。保留所有寄存器，其后，把跳越转移的向量数值放入PC之中）

条件码：E、I、F标志位被置1，其它标志位不变

说明：除系统堆栈指示器外，把所有CPU内部寄存器的内容按照系统堆栈指示器保留在所给的系统堆栈之内，然后跳越转移到以$FFFA地址中高8位，以$FFFB地址中低8位内容为地址的单元之上。

寻址方式：固有（绝对间址）

• **SWI2** 软件中断2（SoftWare Interrupt 2）

记忆符：SWI 2

动作：E标志位置1（保留全部寄存器）

SP′←SP－1，（SP）←PCL
SP′←SP－1，（SP）←PCH
SP′←SP－1，（SP）←USL
SP′←SP－1，（SP）←USH
SP′←SP－1，（SP）←IYL
SP′←SP－1，（SP）←IYH
SP′←SP－1，（SP）←IXL
SP←SP－1，（SP）←IXH
SP←SP－1，（SP）←DPR
SP←SP－1，（SP）←ACCB
SP←SP－1，（SP）←ACCA

SP′←SP－1，(SP)←CCR

PC′←($FFF4)：($FFF5)

(E标志位置1，保留全部寄存器后，把跳越转移的向量地址数值放入PC之中)

条件码： E标志位被置1，其它标志位不变

说明： 除系统堆栈指示器外，把全部CPU内部寄存器的内容保留在系统堆栈指示器所给的系统堆栈之中。然后跳越转移到以$FFF4地址内容为高8位，$FFF5地址内容为低8位的地址之中，SWI2指令由于是结束用户程序的指令。在销售的软件中不应使用该指令。

寻址方式： 固有（绝对间址）

• **SWI3** 软件中断3 (SoftWare Interrupt 3)

记忆符： SWI3

动作： E标志位置1（保留全部寄存器）

SP′←SP－1，(SP)←PCL

SP′←SP－1，(SP)←PCH

SP′←SP－1，(SP)←USL

SP′←SP－1，(SP)←USH

SP′←SP－1，(SP)←IYL

SP′←SP－1，(SP)←IYH

SP′←SP－1，(SP)←IXL

SP′←SP－1，(SP)←LXH

SP′←SP－1，(SP)←DPR

SP′←SP－1，(SP)←ACCB

SP′←SP－1，(SP)←ACCA

SP′←SP－1，(SP)←CCR

PC′←($FFF2)：($FFF3)

(E标志位置1，保留全部寄存器之后，把跳越转移的向量地址数值放入PC之中)

条件码： E标志位被置1，其它标志位不变

说明： 除系统堆栈指示器外，全部CPU内部寄存器保留在系统堆栈之中，然后跳越转移到以$FFF2地址内容为高8位、以$FFF3地址内容为低8位的地址之中)

寻址方式： 固有（绝对间址）

• **SYNC** 和外部事件同步 (SYNChronize to external event)

记忆符： SYNC

动作： 停止指令执行

条件码： 全无变化

说明： 如果执行SYNC指令，CPU为“同步状态”，即停止执行指令，而进入等待中断的状态。如果发生了中断，则清除同步状态，继续执行指令。若处于同步状态之中，可以进行中断，而且中断信号要持续3个机器周期以上，CPU才执行中断程序。但是下面情况不能中断，即中断信号小于3个机器周期时，CPU就不会接受中断，而继续执行指令（也不进行保留寄存器）。在同步状态时，地址总线和数据总线都

将成为高阻抗状态。

寻址方式：固有

注释：该指令是为了在硬件和软件之间进行同步而设置的。例如，在高速数据采集输入中有如下程序

```
                              ≀
FAST  SYNC                 等待数据
中断
LDA DISC                   装入DISC数据并清除中断
STA ,X+                    存在缓冲区
DECB                       数据数目计数，结束否？
BNE FAST                   没有结束时，返回FAST
     ≀
```

同步状态时由于发生了中断，而清除该状态。否则，如果允许进行中断，那么就会把数据传送搞乱（因此，该指令可检查CPU对于非常状态的反应而加以使用）。所以使用中断屏蔽位的置1状态和SYNC指令，可以在高速数据传送之中使用和中断控制输入输出服务相同的方法进行。

· **TFR** 寄存器之间的数据传送（TransFeR register to register）

记忆符：TFR R1，R2

动作：$R_2 \leftarrow R_1$（使R_1的内容放入R_2之中）

条件码：R_2如果不是CCR，则不变

说明：操作数中的第4～7位定义为R_1，第0～3位定义为R_2，实际数值如下：

0000＝D，（A:B）	1000＝A
0001＝X	1001＝B
0010＝Y	1010＝CCR
0011＝US	1011＝DPR
0100＝SP	1100＝未定义
0101＝PC	1101＝未定义
0110＝未定义	1110＝未定义
0111＝未定义	1111＝未定义

数据传送中，如是8位寄存器，则其它也要是8位寄存器，只有相同字长的寄存器彼此才能互相传送。

寻址方式：固有（寄存器）

· **TST** 测试（TeST）

记忆符：TST

动作：TEMP←M－0（从操作数中减0）

条件码：

H：不变

N：运算结果，第7位为1时，被置1

Z：运算结果，所有各位为0时，被置1

V：被置 0

C：不变

说明：根据操作数的内容，使N标志位和Z标志位发生变化，而V标志位被置0。累加器和存储器中的内容不变。在6800处理器中，C标志位也被置 0

寻址方式：　累加器　直接　变址　扩充　间接扩充　间接变址

注释：对于无符号的二进制数使用了TST指令时，再使用BLO指令或BLS指令就没有意义。所以这样说是因为无符号的二进制数一定是0以上的数值。同时在TST指令之后，使用BHI指令也和BNE指令是相同的。带符号的分支转移指令（参看DEC指令中说明）可以使用。

3.1.3. 硬件指令

所谓6809的硬件指令是指可以改变程序执行顺序的输入控制信号，即硬件中断指令 $\overline{FIRQ}$、$\overline{IRQ}$、$\overline{NMI}$、$\overline{RESET}$。

• **FIRQ**　快速中断请求 (Fast Interrurt ReQuest)

动作：F标志位为 0 时：

SP′←SP－1，　(SP)←PCL

SP′←SP－1，　(SP)←PCH

E标志位置‘0’（只保留PC和CCR）

SP′←SP－1，　(SP)←CCR

F，I标志位置‘1’（禁止后面中断）

PC′←（$FFF6）：（$FFF7）

条件码：除E、F、I标志位以外，不变。

说明：当F标志位被置0态时，如果 $\overline{FIRQ}$ 输入端变为低电平，则在当前执行中的指令结束之后，即进行该中断处理。程序计数器和条件码寄存器内容保留在系统堆栈之后，则跳越转移到以$FFF6地址内容为高8位，以$FFF7地址内容为低8位的地址之中。为返回中断前的程序，使用RTI指令进行，在执行CWAI指令以后，如果发生了FIRQ，除了系统堆栈指示器外，全部CPU内部寄存器的内容被保留起来，而且仍可跳越转移到中断程序上。

注释：因为进入FIRQ处理程序时，自动地置‘1’到I标志位，所以在FIRQ处理程序之中，IRQ的中断即不能发生作用。如果不需要使用优先顺序时，则在FIRQ程序内可使I标志位清零。由于使用FIRQ中断把全部寄存器的内容保留在堆栈中不费什么事，所以用TST/INC/DEC指令等对存储器进行操作。

• **IRQ**　中断请求 (Interrupt ReQuest)

动作：若I标志位为零，

SP′←SP－1，　(SP)←PCL

SP′←SP－1，　(SP)←PCH

SP′←SP－1，　(SP)←USL

SP′←SP－1，　(SP)←USH

SP′←SP－1，　(SP)←IYL

SP′←SP－1，(SP)←IYH

SP′←SP－1，(SP)←IXL

SP′←SP－1，(SP)←IXH

SP′←SP－1，(SP)←DPR

SP′←SP－1，(SP)←ACCB

SP′←SP－1，(SP)←ACCA

置‘1’E标志位（全部寄存器保留）

SP′←SP－1，(SP)←CCR

置‘1’I标志位（禁止后面的IRQ）

PC′←($FFF8)：($FFF9)

条件码：除E、I标志位外不变。

说明：若I标志位被置0，而且$\overline{IRQ}$输入又为低电平，则在当前执行中的指令结束后，进行IRQ中断处理。为了返回原来的程序使用RTI指令。FIRQ可以在IRQ处理程序中产生中断作用，这时，只要IRQ向量地址读入之后，不管何时都可以接受中断。

• **NMI** 非屏蔽中断（Non-Maskable Interrupt）

动作：SP′←SP－1，(SP)←PCL

SP′←SP－1，(SP)←PCH

SP′←SP－1，(SP)←USL

SP′←SP－1，(SP)←USH

SP′←SP－1，(SP)←IYL

SP′←SP－1，(SP)←IYH

SP′←SP－1，(SP)←IXL

SP′←SP－1，(SP)←IXH

SP′←SP－1，(SP)←DPR

SP′←SP－1，(SP)←ACCB

SP′←SP－1，(SP)←ACCA

置‘1’E标志位（全部寄存器保留）

SP′←SP－1，(SP)←CCR

置‘1’I、F标志位（禁止中断）

PC′←($FFFC)：($FFFD)

条件码：除E、F、I标志位之外，不变。

说明：在$\overline{NMI}$输入端输入负沿信号时，在执行中的指令结束时，除系统堆栈指示器之外，全部CPU内部寄存器的内容都被保留在系统堆栈之中。接着跳越转移到由NMI跳越转移向量地址所给的地址之中。如果在$\overline{NMI}$输入端连续地输入负沿，则可以连续进行NMI处理。NMI处理可以因RESET输入而在内部被终止，此后即使$\overline{NMI}$负沿被锁存下来，但只要在堆栈指示器中没输入数据就不会动作。

• **RESTART** 再启动

动作：CCR′←X1X1XXXX（X：0或1）

DPR′←$00

PC′← ($FFFE) : ($FFFF)

说明：为了开始执行程序，CPU 如果被初始化（在电源加上时），则CCR的I标志位和E标志位被置 1，在DPR 中输入 $00，在 PC 中输入 $FFFE，$FFFF地址中的内容。

6809 中断保留和恢复的顺序如图 3.1 所示。

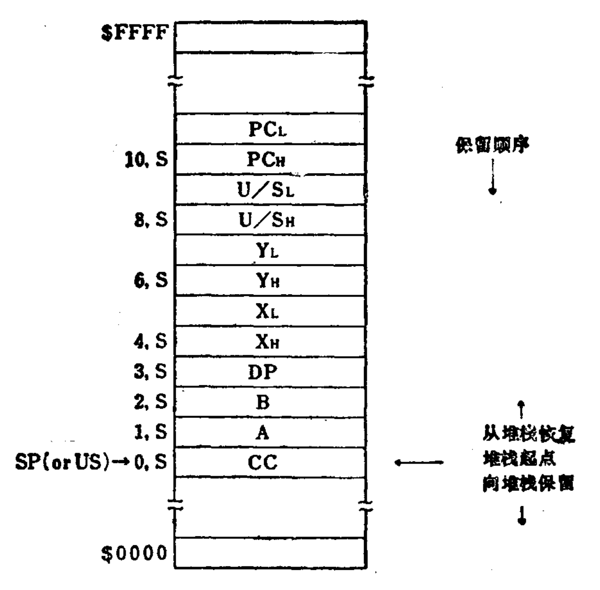


图3.1 6809保留/恢复、中断保留的顺序

3.1.4 后缀字节

在6809中，接在指令（操作码）后的后缀字节是操作码的修饰字节，作为指定寄存器使用。

在变址寻址方式中，也要使用接于操作码后的后缀字节来给定变址寻址中更详细的方式。因此，这里要对有关指定寄存器和变址寻址的后缀字节加以说明。

1. TFR、EXG指令

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

TFR、EXG指令的后缀字节为分为高 4 位b_7、b_6、b_5、b_4和低 4 位b_3、b_2、b_1、b_0两个半字节。高 4 位半字节（$b_{7、6、5、4}$）指定源寄存器，低 4 位半字节（$b_{3、2、1、0}$）指定目的寄存器。

TFRX，Y指令中，数据为X→Y传送。这时X为源寄存器，Y为目的寄存器。

数字位的结构分配，源、目的寄存器的指定方式也相同，如下：

b_7 b_6 b_5 b_4

b_3 b_2 b_1 b_0

0000 累加器（D）

0001 变址寄存器（X）

0010 变址寄存器（Y）

0011 用户堆栈指示器（U）

0100 系统堆栈指示器（S）

0101 程序计数据（PC）

1000 累加器（A）

1001 累加器（B）

1010 条件码寄存器（CC）

1011 直接页面寄存器（DP）

b_7、b_3表示寄存器的长度，0时为16位长的，1时为 8 位长的。

除上述之外的数值寄存器是不确定的，指定的是无效寄存器。

2. PSH、PUL

6809的PSH、PUL指令的后缀字节对用户堆栈指示器（U）和系统堆栈指示器（S），有一项不同。但不管使用哪个堆栈指示器时，按照数字位安排的组合可以指定所有的寄存器。因此每次用一条指令即可把8个寄存器的内容压入或者弹出存储器堆栈，如图3.2。

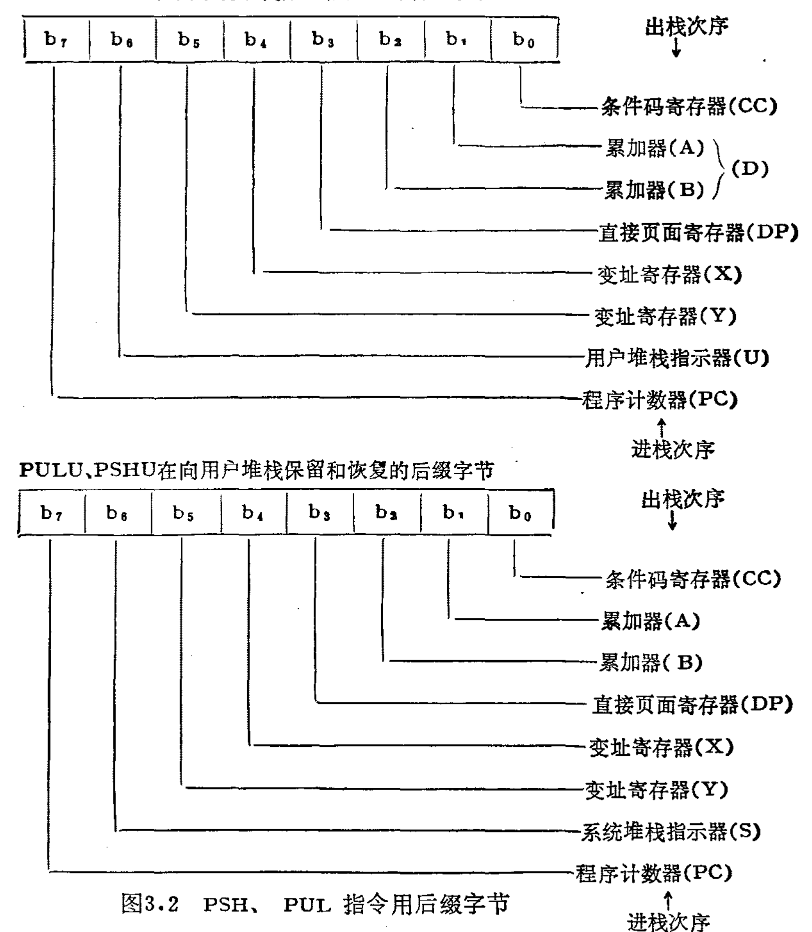


图3.2 PSH、PUL 指令用后缀字节

当用PSHU或PULU时，后缀字节的第6位相当于S寄存器在进栈或出栈。当用PSHS或PULS时，后缀字节第6位相当于U寄存器。如果只是部分寄存器进栈，其次序也是相同的。16位的寄存器进栈时，低位字节在前，高位字节在后。16位寄存器出栈时，高位字节在前，低位字节在后。全部6809的内部寄存器进栈时需要12个字节的堆栈存储器。累加器D不算，因为它是由累加器A和B串接而成的。

3. 变址等寻址方式

在变址寻址方式中，后缀字节具有重要的作用。另外在用程序计数器相对寻址时，也需要后缀字节。总之后缀字节的作用是：

（1）后缀字节可规定以下寻址方式：

程序计数器相对寻址

零偏值变址寻址

常数偏值变址寻址

累加器偏值变址寻址

自动增/减变址寻址

（2）后缀字节可规定程序计数器、X、Y、S、或U寄存器作为指示寄存器；

（3）后缀字节可规定程序计数器相对寻址中偏值数值大小和常数偏值变址寻址；

（4）后缀字节可规定在累加器偏值变址寻址中所用的累加器（A、B或D）；

（5）当使用带符号的5位常数偏值时，偏值数值要作为后缀字节一部分来考虑。

以上所有情况如何用1字节的信息格式来实现？其后缀字节的格式如图3.3所示。后缀字节分为以下几部分：

寻址方式部分（0～3位）

间接选择部分（第4位）

指示寄存器部分（第5、6位）

5位偏值部分（第7位）

BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0

指示寄存器选择

寻址方式选择

5位偏值选择

间接选择

图3.3　变址寻址的后缀字节格式

现在说明上述后缀字节的每一部分情况。

寻址方式部分（0～3位）

后缀字节的低4位决定寻址方式。寻址方式和有关各位的内容如表3.1所示。其中R代表指示寄存器（X、Y、S、或U）。变址寻址方式和程序计数器相对寻址方式都可由后缀字节低4位来规定。同时还多一种间接扩充寻址方式也予以规定。

表3.1　寻址方式部分数字位内容规定

后缀字节寻址方式部分 3位	2位	1位	0位	寻址方式	符号
0	0	0	0	自动加1	R+
0	0	0	1	自动加2	R++
0	0	1	0	自动减1	−R
0	0	1	1	自动减2	−−R
0	1	0	0	零偏值	R± 0
0	1	0	1	ACCB偏值	R±ACCB
0	1	1	0	ACCA偏值	R±ACCA
1	0	0	0	8位带符号偏值	R± 7位
1	0	0	1	16位带符号偏值	R±15位
1	0	1	1	ACCD偏值	R±ACCD
1	1	0	0	PC相对——8位带符号	PC± 7位
1	1	0	1	PC相对——16位带符号	PC±15位
1	1	1	1	间接扩充	[n]

间接选择部分（第4位）

后缀字节的第4位是间接寻址的选择位。该位为0时表示用直接寻址。这样实际上操作数中安排了操作中规定的地址。该位为1时表示用间接寻址。其中该地址代表包括实际操作数地址的一个单元。

指示寄存器部分（5、6位）

该部分在确定地址时选择所用的指示寄存器。指示寄存器与数字位内容的关系如表3.2所示。这四个可变址的寄存器（X、Y、S、U）任何一个都可作为指示寄存器。

5位偏值部分（第7位）

除要求带符号的5位偏值以外，该位总处在逻辑1态。当第7位为0时，则后缀字节中的第0～4位将为5位2的补数偏值。第4位为符号位，第0～3位表示偏值的权数。

表 3.2 指示寄存器数字位内容规定

后缀字节指示寄存器部分		指示寄存器
6位	5位	
0	0	R=X
0	1	R=Y
1	0	R=U
1	1	R=S

变址寻址方式中后缀字节规定的所有寄存器的定义如表3.3所示。

表 3.3 变址寻址方式后缀字节各位的意义

后缀字节寄存器位								偏值种类	变址寻址方式
7	6	5	4	3	2	1	0		
0	R	R	X	X	X	X	X	5位偏值	EA=,R± 4位偏值
1	R	R	0	0	0	0	0	0偏值,自动加1	,R+
1	R	R	I	0	0	0	1	0偏值,自动加2	,R++
1	R	R	0	0	0	1	0	0偏值,自动减1	,−R
1	R	R	I	0	0	1	1	0偏值,自动减2	,−−R
1	R	R	I	0	1	0	0	0偏值	EA=,R± 0偏值
1	R	R	I	0	1	0	1	ACCB偏值	EA=,R±ACCB偏值
1	R	R	I	0	1	1	0	ACCA偏值	EA=,R±ACCA偏值
1	R	R	I	1	0	0	0	8位偏值	EA=,R± 7位偏值
1	R	R	I	1	0	0	1	16位偏值	EA=,R±15位偏值
1	R	R	I	1	0	1	1	ACCD偏值	EA=,R±D偏值
1	X	X	I	1	1	0	0	8位偏值(PC)	EA=,PC± 7位偏值
1	X	X	I	1	1	0	1	16位偏值(PC)	EA=,PC±15位偏值
1	R	R	1	1	1	1	1	间接扩充寻址	EA=,Add.

变址寻址方式后缀字节的详细表格说明如表3.4所示。

表 3.4 变址寻址方式的后缀字节

偏值种类			直接				间接			
			汇编形式	后缀字节	+~	+#	汇编形式	后缀字节	+~	+#
零			,R	1rr00100	0	0	[,R]	1rr10100	3	0
零	自动操作	自动加 1	,R+	1rr00000	2	0	—	—	—	—
零	自动操作	自动加 2	,R++	1rr00001	3	0	[,R++]	1rr10001	6	0
零	自动操作	自动减 1	,−R	1rr00010	2	0	—	—	—	—
零	自动操作	自动减 2	,−−R	1rr00011	3	0	[,−−R]	1rr10011	6	0
常数		5 位	N,R	0rrnnnn	1	0	—	—	—	—
常数		8 位	N,R	1rr01000	1	1	[N,R]	1rr11000	4	1
常数		16位	N,R	1rr01001	4	2	[N,R]	1rr11001	7	2
累加器		Acc A	A,R	1rr00110	1	0	[A,R]	1rr10110	4	0
累加器		Acc B	B,R	1rr00101	1	0	[B,R]	1rr10101	4	0
累加器		Acc D	D,R	1rr01011	4	0	[D,R]	1rr11011	7	0
程序计数器相对		8 位	N,PCR	1××01100	1	1	[N,PCR]	1××11100	4	1
程序计数器相对		16位	N,PCR	1××01101	5	2	[N,PCR]	1××11101	8	2
间接扩充寻址方式			—	—	—	—	[N]	1××11111	5	2

R=X、Y、U、S　　X=任意　　rr: 00=X
01=Y
10=U
11=S

+~, +# 分别表示各条指令的基本周期数和被加在基本字节数上的数值

在 8 位常数偏值方式中，接在后缀字节上的为 1 字节操作数；在16位常数偏值方式中，接在后缀字节上的为 2 字节操作数。

在用指令写汇编码之前，需要使用代表各种后缀字节的汇编码，该汇编码如表3.5所示。

表 3.5　后缀字节汇编码

寻址方式符号	汇编码
R±0	,R
R±4 位	n,R
R±7 位	n,R
R±15位	n,R
R±ACCA	A,R
R±ACCB	B,R
R±ACCD	D,R
PC±7 位	n,PCR
PC±15位	n,PCR
R+	,R+
R++	,R++
−R	,−R
−−R	,−−R

在该表中，n表示偏值大小，R为指示寄存器，PC为程序计数器。例如LDA 23,X 含意是：把常数偏值23_{16}加到指示寄存器X上所代表的存储器单元中的内容装入累加器A。LDX D,Y 含意是：把累加器D的内容加到指示寄存器Y上所代表的存储器单元中的内容装入变址寄存器X。

为了确定汇编码和后缀字节的内容，再举几例说明：

例 1：设把累加器D的内容存储在存储单元中，其地址是S指示寄存器使用偏值为－5。求表示该操作的汇编码？

从表3.5中可知汇编码将是STD－5，S。执行该操作时后缀字节的内容将是：

$$7B_{16} = 01111011_2$$

让我们分析一下为什么如此?因为常数偏值为－5，可用带符号的5位偏值。所以偏值为后缀字节的一部分。后缀字节的第7位为0，表示为5位常数偏值。第5、6位规定S寄存器为指示寄存器。第0～4位代表常数偏值，第4位表示为负偏值需为1，第0～3位为2的补数偏值(1011)。

例 2：设后缀字节为A0，说明其含意，并确定使用该后缀字节指令所采用的汇编码。

参看表3.3：后缀字节 = $A0_{16} = 10100000_2$

第7位为1，意思是不是5位常数偏值。因为第5、6位为01，所以指示寄存器为Y寄存器。第0～4位的寻址方式为全0，表示为自动加1的寻址方式。该汇编码需要的后缀字节将是 ,Y+。用该后缀字节操作的有效地址将是Y变址寄存器的内容。在操作执行之后，Y变址寄存器的内容将自动＋1。

例 3：如果后缀字节是$ED_{16} = 11101101_2$。参看表3.3的解释：寻址方式为1101时，是带符号的16位偏值的计数器相对寻址。间接部分为0，即不是间接寻址。指示器部分两位都是1，但因为是PC相对寻址，可不考虑该两位内容，或者认为该两位内容所表示的指示寄存器为程序计数器。最后5位常数偏值部分为1，说明不是5位偏值的情况。

这时汇编码将为：n，PCR，其中n为带符号的16位偏值。那么该操作的有效地址即为常数偏值n被加到当前程序计数器内容之上的数值。

例 4：设要求装入X变址寄存器，其内容是X指示寄存器采用累加器D作为常数偏值时的存储器单元中的内容。求代表该操作的汇编码和后缀字节？

从表3.5可知汇编码应是：LDX D，X。因为要求是累加器D作偏值，从表3.3可知后缀字节的寻址方式一定是1011。还因规定X为指示寄存器，故指示寄存器部分一定是00，间接寻址部分（第5位）必须是0，而且5位常数偏值部分（第7位）必定是1。所以后缀字节应是10001011_2或$8B_{16}$。

3.1.5 子程序调用

6809的子程序调用有以下七种寻址方式。

直接寻址方式	间接扩充寻址方式
扩充寻址方式	间接变址寻址方式
变址寻址方式	8位相对寻址方式
	16位相对寻址方式（长相对）

在分支转移子程序之前，首先应把程序计数器内容保留到系统堆栈区（S）之中，因为从子程序恢复时要利用这些内容。保留程序计数器，更新系统堆栈指示器S之后，对跳越转移

的目的地址的计算方法，按各种寻址方式有所不同。

在直接寻址方式中，直接页面寄存器的内容为有效跳越转移地址的高位字节，一字节数据的操作数为低位字节。

在扩充寻址方式中，2字节数据的操作数为有效跳越转移地址。

在相对寻址方式中，要把8位或16位的偏值加在程序计数器的数值之上作为有效地址。

从子程序调用的地址到子程序的距离，在+127～-128字节以内的为8位偏值方式(BSR)，在其以上的用LBSR指令。

绝对寻址方式（直接寻址、扩充寻址）当然不能使用在位置独立的程序之中。

在子程序内再调用以后的子程序的次数（嵌套级数），原理上讲，只是使所用的系统堆栈区增大以外，没有什么限制。每用一级嵌套使用二个字节，若系统堆栈区有40个字节时，则用中断等方式使用系统堆栈为20级，在6809中没有条件子程序调用。

其次，从子程序返回到主程序时，一般情况使用子程序返回的指令RTS。

RTS指令是把保留在系统堆栈内的恢复地址，返回到程序计数器的一种跳越转移指令。执行该指令时，相当于程序计数器进行PULS PC。如果从子程序返回时，同时需要恢复寄存器内容，例如：

```
PULS A, B
RTS
```

在从子程序结束返回时，用PULS A,B,PC一条指令即可解决问题。这种PULS指令,后缀字节为二个字节；如果用PULSA,B和RTS不仅节约一个字节,而且执行时间也可缩短。

从子程序返回时，对PULS PC和RTS选择哪一个，这要取决于所保留的寄存器是否也要返回。

分支转移子程序的用法在此要加以说明。在分支转移子程序中，有8位偏值（BSR）和16位偏值（LBSR）两种。

这两种分支转移子程序可在作位置独立程序时使用，或者在作结构化程序时使用。在实现长相对分支转移子程序调用（LBSR）时，没有6800中那种从-128～+127字节分支转移目的地址范围的限制，而是在整个64K字节的空间使用相对寻址方式都可以对子程序进行调用。

3.2 寻址方式

6809是一个高级的8位微处理器。但6809作为表示一个处理器能力强弱的指令系统所拥有指令数不是很多，其最重要的考虑是：对处理器本身可以使用基本指令有多少种不同的方法。指令用不同方法对数据进行访问和操作的根本手段是采用不同的寻址方式。6809有59条基本指令，可以使用10种寻址方式，操作总数可达1464种。6809的10种寻址方式是：固有寻址（隐含）、立即寻址、直接寻址、扩充寻址、分支相对寻址、变址寻址、间接扩充寻址、程序计数器相对寻址、间接变址寻址和寄存器寻址。

实际上这10种基本寻址方式变化后使总寻址方式有19种。例如一种功能最强的寻址方式

是变址寻址，它有五种选择：零偏值、常数偏值（5、8、16位）、累加器偏值（A、B、D）和自动加或自动减（1或2）。而所有这些选择采用间接变址时又都可以间接地访问数据。间接寻址意思就是被寻址的存储器单元之中存的是操作数的地址而不是操作数本身。因此指令先找操作数地址再去寻找操作数。间接操作在扩充和程序计数器相对寻址方式中也可以使用。

和6800一样，分支转移的操作使用相对寻址。但6809也可用相对寻址对存储器中数据进行访问和操作。而且还有短相对（8位偏值）和长相对（16位偏值）两种类型。所以6809的程序完全可以作成位置独立式的。

最后，使用寄存器寻址方式可以在6809内部任何两个长度相同的寄存器之间传送和交换数据。

一个8位微处理器怎么能执行这么多的操作？一条指令怎样规定寻址方式？其原因如前所述，这就是因为使用了后缀字节。在变址、间接变址、间接扩充、程序计数器相对和寄存器寻址中都可以使用后缀字节。在指令语句中后缀字节接在指令操作码之后。用它规定寻址方式和操作中所用的内部寄存器。在寄存器寻址方式时，后缀字节可以使两个长度相同的寄存器传送和交换数据。

3.2.1 何谓寻址方式

寻址方式要对按记忆符所表示的基本指令进行修改，目的是决定怎样解释进行操作的数据（操作数）。虽然把基本指令和操作数交给计算机，但在执行时怎样使用操作数？如果没有把寻址方式告诉计算机，计算机也就不能正确地执行指令。例如给出操作数的数值是1000，所谓1000是指1000地址呢？还是指在1000地址中所存的数据呢？这一点是不清楚的。所以寻址方式可以对这种不确定的操作数的数据给以明确处理。

例如，把数据装入累加器这一指令。作为最一般的方法是把被装数据所在的存储器地址规定为操作数（直接方式、扩充方式）。同时，还有把数据本身直接作为操作数的方法（立即方式），还可有取变址寄存器内容和操作数之和来作为存储器地址的方法（变址方式）以及被装入数据所在的存储器地址存在存储器中的地址来作为操作数的方法（间接方式）等等。

所谓寻址方式就是表示指定这些地址之中为哪一种的方法，即6809具有上述所说的那些寻址方式。

这些寻址方式不是对所有指令都能使用，而是根据指令的不同，有的方式可以使用，有的方式不能使用。

固有寻址方式和累加器寻址方式都是没有操作数的方式，所以属于这种方式的指令就不能用其它方式，

寄存器寻址方式是根据后缀字节来指定寄存器的指令，只有TFR、EXG、PSH、PUL这四种指令符合，但这些指令不能使用其它寻址方式。

相对寻址方式，只是在分支转移指令中采用的一种方式。

其余六种寻址方式，多数指令都可以使用，只有立即寻址方式不用的指令占多数。也有只使用立即寻址方式的指令。使用变址寻址方式的指令，全部都可以使用间接扩充寻址和间接变址寻址方式，反之亦然。这三种方式只有根据后缀字节加以区别，操作码是一样的。但

也有只使用这三种寻址方式的指令。

按照寻址方式，处理和计算操作数本身数据和寄存器的内容，就会形成所要求的对象地址。通常称该地址为有效地址（Effective Address），简称EA。

对各种寻址方式的说明，将使用汇编语言中的一部分符号和描述方法。因此，需预先说明有关公用的数据种类。

使用术语符号的说明：

数据种类（同汇编语言）

二进制数%01001101	在数据之前写%
八进制数@735	在数据之前写@
十进制数 168	数值前什么不写
十六进制数$370	在0～9和字母之前写$

符号种类

:=	把右边的内容代入左边	-	减法
=	比较右边和左边内容	∩	逻辑乘法（AND）
*	乘法	∪	逻辑加 （OR）
+	加法	n	执行顺序

3.2.2 立即寻址方式

这种寻址方式不是指定存储器的地址，而是把操作数本身规定的内容作为运算处理的对象。所以这种寻址方式在指令语句中包括数据本身这种操作数。利用累加器和变址寄存器进行操作的加、减、装入、与、或、比较等指令都可使用立即寻址方式。

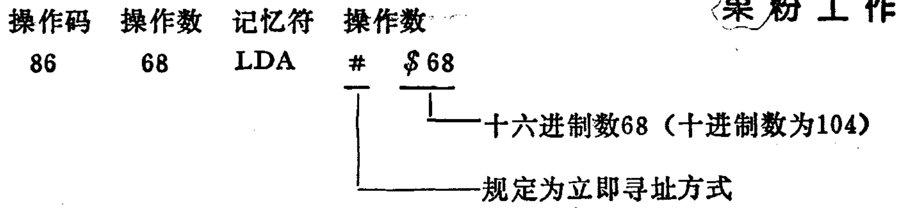
操作数的长度（字节数）由操作码来决定。当立即操作包括累加器A或B时，指令将为2字节，操作码一个字节，操作数一个字节。当操作包括16位寄存器，如累加器D或变址寄存器X、Y、S、U之一时，指令将为3或4字节，操作码1或2个字节（取决于指令本身），操作数2个字节。在6800中操作码永远不会超过一个字节的长度，然而在6809中会经常遇到2个字节指令的操作码。实质上这是因为6809有256种以上可执行的操作指令。而且当用16位寄存器时，也需要2字节的操作数。立即寻址格式如图3.4所示。

指 令 操 作 码 （1或2字节）
操作数（数据） （1或2字节）

图3.4 立即寻址格式

作为立即寻址方式所使用的指令有LDA、LDX、ADDA、SBCB、ANDA等。如ORCC等指令只有这种方式。

例：往累加器A装入数据



其工作原理如图3.5所示。

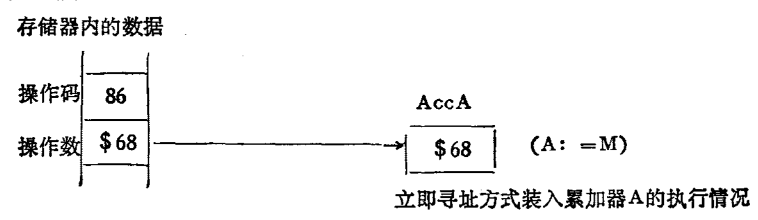


图5.5　立即寻址方式之例

3.2.3　固有寻址方式

固有寻址方式又称隐含寻址方式，是最简单的寻址方式，因为它只有一字节，根据操作码即可确定寻址，而操作数是预先被决定的。

例如乘法指令（MUL），运算对象只限于ACCA和ACCB，并不能指定别的寄存器或存储器。因此可以认为这种寻址方式是不可选择的。

处理对象之所以定为寄存器或累加器，这是因为考虑只设置操作码即可作为一种指令的方法，所以固有寻址方法又称为没有操作数指令的寻址方式。

6809设有执行累加器和寄存器间的传送、运算等几种指令，因这些指令都要按照后缀字节来指定寄存器，所以不算固有寻址这一类，而是作为寄存器寻址方式。

属于累加器的操作指令，如加1、减1、清零、移位（左移或右移）、变补等操作都算这种寻址方式。当指令是在一个累加器中进行操作时，这种寻址方式也可称作累加器寻址。如：CLRA、CLRB、ASRA等指令的操作也作为固有寻址的一部分。

固有寻址方式的指令有ABX、DAA、SWI、ASRA等。

3.2.4　寄存器寻址方式

在6809中还有一种主要寻址方式即寄存器寻址方式。因为在6809中任何一个用户寄存器的内容都可以传送到任何其它寄存器，或者与其它任何寄存器进行交换，只要长度相同。所以寄存器寻址方式是以6809内的寄存器的内容作为操作数。使用固有寻址指令即可完成这个任务，所以寄存器寻址可为固有寻址方式的一部分。但需要后缀字节来规定所含的寄存器。所以在内部寄存器之间进行数据传送或交换的任何指令都称为寄存器寻址指令。这种指令为2字节长，由指令操作码后跟后缀字节构成。指令和后缀字节格式如图3.6所示。后缀字节分为两部分：源寄存器部分和目的寄存器部分。数据可以从源寄存器传送到目的寄存器，或者在源和目的寄存器之间进行交换。每个内部寄存器的数字位编码如图3.6所示。其中只有一个要求就是两个寄存器的长度要相同。

6809寄存器寻址方式的基本指令有TFR、EXG、PSH、PUL等4种指令。

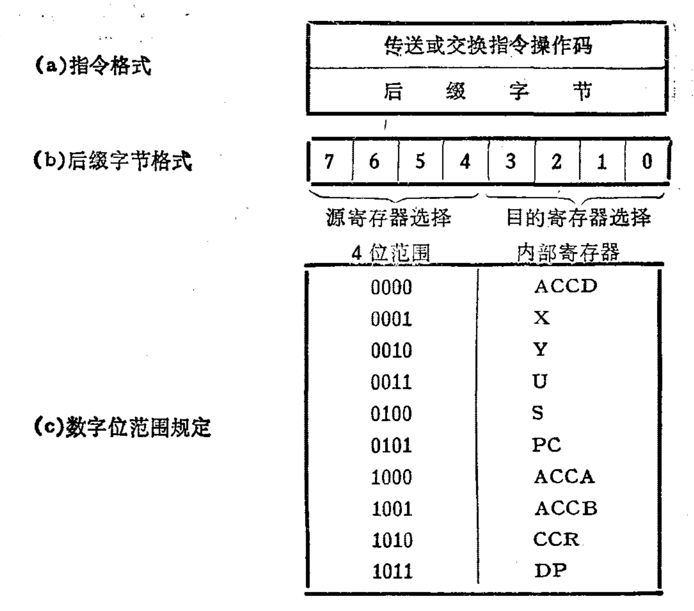
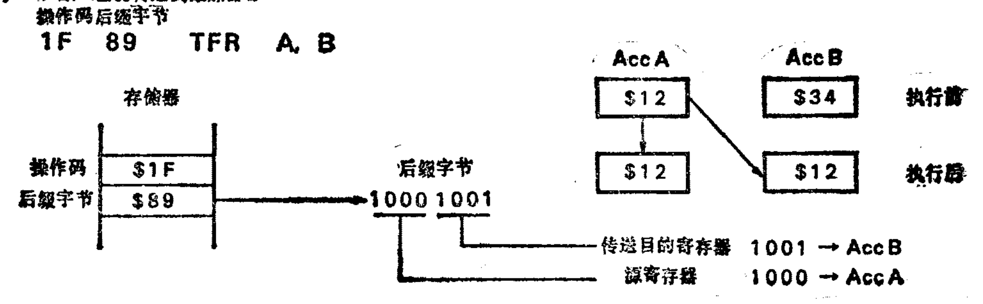


图 3.6　寄存器寻址方式

寄存器寻址方式之例如图3.7所示。



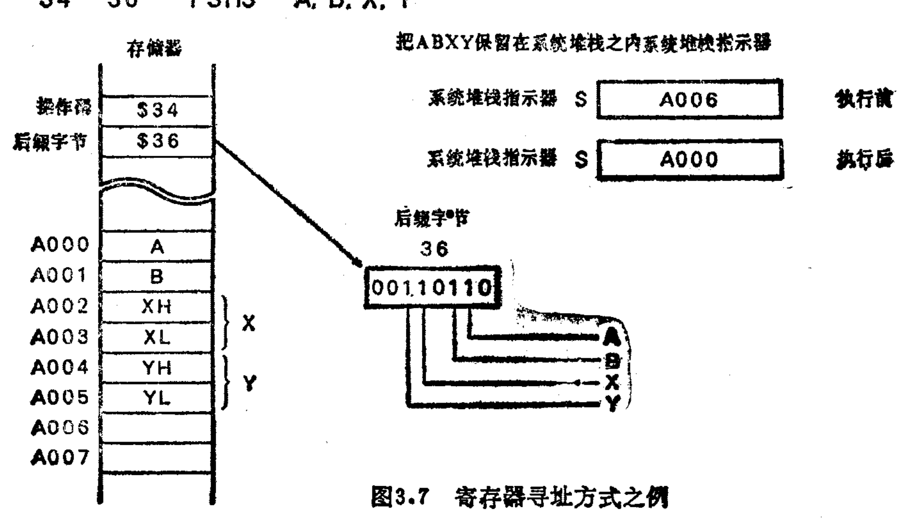


图3.7　寄存器寻址方式之例

3.2.5 绝对寻址方式

绝对寻址方式中的操作数所表示的地址即为有效地址。也就是把作为对象的数据所在的存储器地址直接表示的寻址方式。

6809的绝对寻址方式中，有直接寻址方方式、扩充寻址方式、间接扩充寻址方式三种。

1. 直接寻址方式

6809直接寻址方式克服了6800只能访问256个存储单元（0000～00FF）的缺点，而考虑使直接页面寄存器作为6809直接寻址结构的一部分。直接页面寄存器实际为有效地址的高位字节。指令中的操作数为低位字节。所以在直接寻址方式中，操作码后的操作数为 8 位，作为16位存储器地址的低 8 位使用。而高 8 位要看直接页面寄存器的内容。

DPR的内容为 0 0 时，可称为访问 0 页面；DPR为01时，可访问 1 页面；直到有255 (FF)页面。所以DPR可以访问256页的存储区。每页面有256个地址字节，换句话说，6809的全部地址空间都可以用DPR按页寻址。因为每页256字节，256页共有64K字节的空间。DPR的用途是在那些经常是全变量进行访问的高级语言之中。所谓全变量的意思就是该数值可以访问整个程序。相反局部变量其数值只可在所定义的程序块（或子程序）中进行访问。在子程序中，可用DPR指到包括全变量的某一页面，以及包括局部变量的堆栈。只要语言编译程序注意考虑DPR的数值，就不会带来任何问题。DPR还可用于多任务操作之中。多任务指的是在一个程序中有几个分开的但是相关任务的操作。在这种应用中，每个任务都用程序分配给不同的页面，并经DPR进行访问。当写程序时应仔细考虑使用DPR的问题，因为经常不知其数值为何值。一旦用好DPR则会极大提高效率节约程序的字节数。

在6809执行RESET后，DPR为全 0 （页面 0 ），这时6809的直接寻址即同6800的直接寻址。

直接寻址方式所使用的指令有LDA、ADDD、TST、STA、ANDA等。

直接寻址方式之例如图3.8所示。

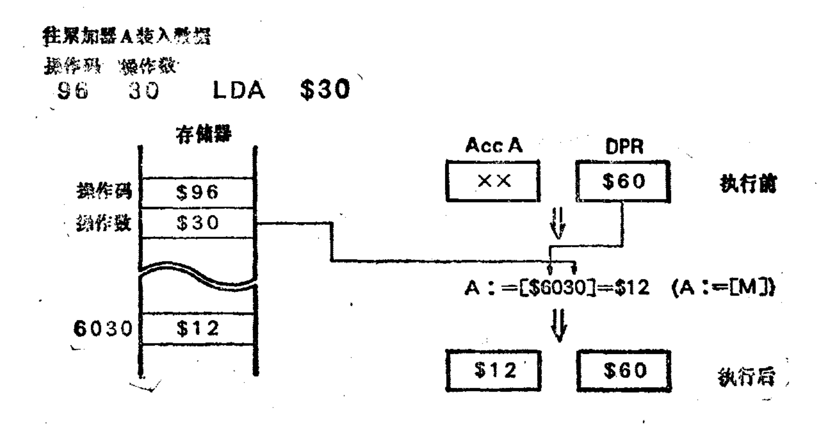


图 3.8 直接寻址方式之例

2. 扩充寻址方式

扩充寻址方式是指接在操作码后的操作数表示的是64K字节的存储器空间，使用16位有效地址。所以地址为 2 字节长，有高地址字节和低地址字节。指令操作码根据不同的指令有

1或2字节。因此用扩充寻址方式的6809指令为3或4字节。扩充寻址方式的指令格式如图3.9所示。扩充寻址方式之例如图3.10所示。

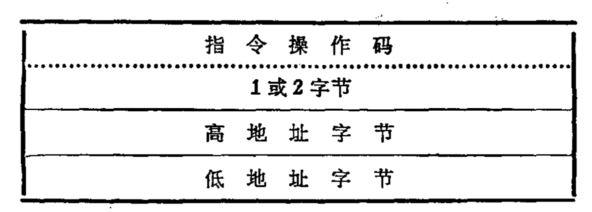


图 3.9 扩充寻址方式格式

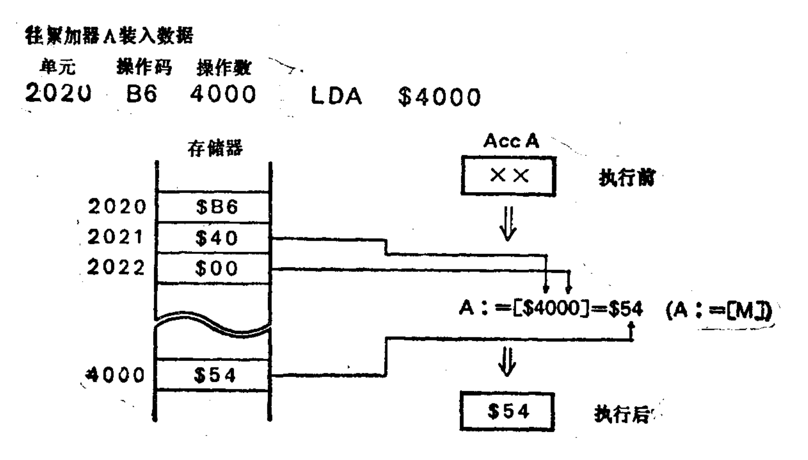


图 3.10 扩充寻址方式之例

因为操作数是用16位有效地址来表示，所以该有效地址所定的存储器单元在位置上不变，因此这种寻址方式称为绝对寻址方式。

扩充寻址方式，显然在编制位置独立程序时不能使用。

这种方式能使用的指令有LDA、ADDD、STX、INC、LSL、TST等。

3. 间接扩充寻址方式

在扩充寻址方式中操作数直接是有效地址。而在间接扩充寻址方式中，由操作数所表示的地址是存储有效地址的单元。也就是说，操作数所代表的存储器地址中的内容是有效地址中的高8位地址；下一个存储器地址中的内容是有效地址中的低8位地址。

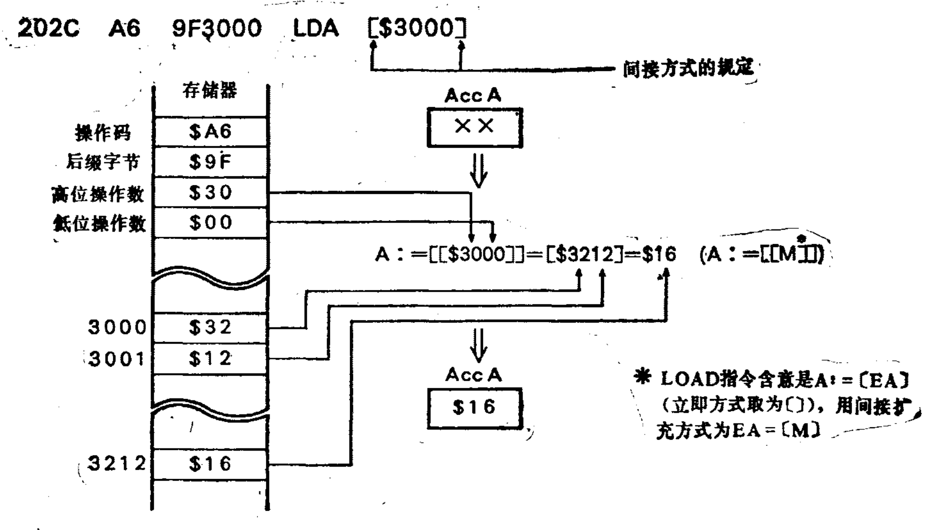
6809中的间接扩充寻址方式和变址寻址方式的操作码相同，它们之间的区别用后缀字节的数字位的结构决定。由表3.3知，后缀字节的间接寻址范围必须是1111。不用指示寄存器决定有效地址，而是指示寄存器所表示的范围须被清0（见表3.4）。间接寻址部分（第4位）为1，用以表示间接寻址，而表示5位偏值部分的第7位须为1。所以后缀字节的内容应该是%10011111（$9F）。

间接扩充寻址方式之例如图3.11所示。

间接扩充寻址方式所用的指令有LDA、LDX、STU、JMP等。

3.2.6 相对寻址方式

6809有两种相对寻址方式，它们是分支转移相对寻址和程序计数器相对寻址。用这种相



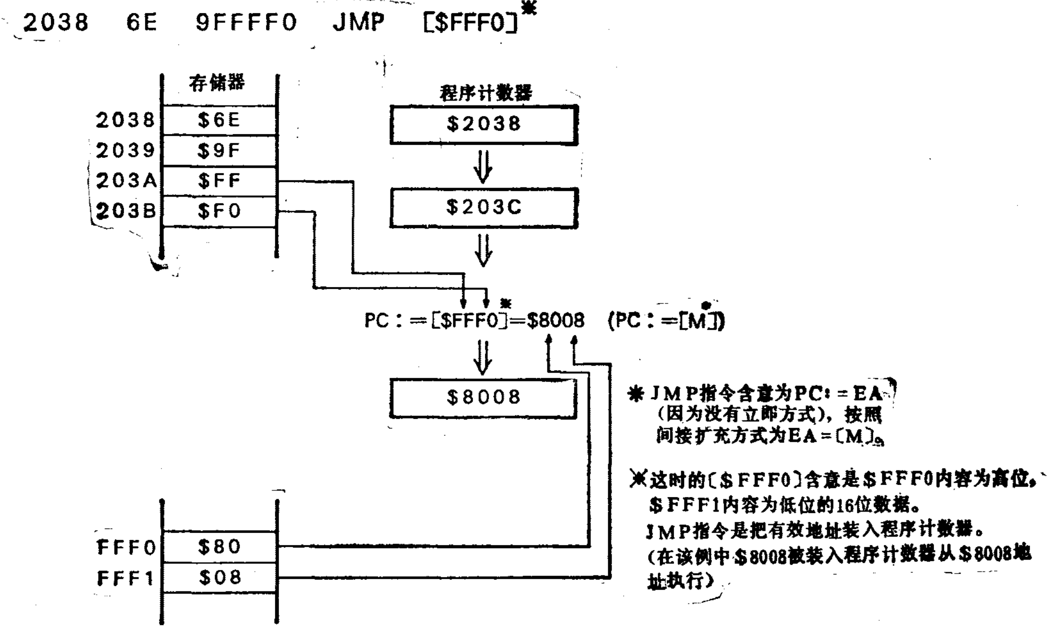


图3.11 间接扩充寻址方式之例

对寻址方式时可以写出6809的完全位置独立的程序。

1. 分支转移相对寻址

分支转移指令操作码为1字节或2字节，其后可能有1字节或2字节的操作数，该数为2的补数（包括正、负二进制数），加在程序计数器PC之上即为分支转移相对寻址。

分支转移指令分有条件的和无条件的二类。有条件的分支转移，只有在分支转移条件被满足时，操作数的偏值才加在PC值之上。满足条件时可用下式表示：

PC：=PC+符号·偏值

偏值为1字节时称为短分支转移，2字节时称为长分支转移。6809的长、短分支转移的

指令格式如图3.12所示。

分支转移指令操作码（1字节）
相对地址偏移（1字节）

（a） 短分支转移指令格式

分支转移指令操作码（第一字节）
分支转移指令操作码（第二字节）
相对地址偏值（高位字节）
相对地址偏值（低位字节）

（b） 长分支转移指令格式（无条件长分支 转 移和子程序长分支转移除外）

图 3.12 6809分支转移指令格式

为了转移到目的地址所需的 8 位相对地址偏值的确定方法见表3.6示出的6809短分支 转移的计算表。这种分支转移有一个严重的限制，就是因为带符号的偏值只有一个字节，用字节的最高位决定转移方向（向前或向后），转移地址限制在-128_{10}～$+127_{10}$之间。为 了 超过这个范围可以分支转移到一个分支点或用跳转指令。但是使用跳转指令时，就不能实现位置独立程序，因为这时用了绝对地址。然而如果相对地址偏值为 2 字节，则可实现转移范围为

表 3.6 6809短分支转移计算表

MSH·B	F	E	D	C	B	A	9	8	
LSH·B									LSH·F
·	·	16	32	48	64	80	96	112	0
F	1	17	33	49	65	81	97	113	1
E	2	18	34	50	66	82	98	114	2
D	3	19	35	51	67	83	99	115	3
C	4	20	36	52	68	84	100	116	4
B	5	21	37	53	69	85	101	117	5
A	6	22	33	54	70	86	102	118	6
9	7	23	39	55	71	87	103	119	7
8	8	24	40	56	72	88	104	120	8
7	9	25	41	57	73	89	105	121	9
6	10	26	42	58	74	90	106	122	A
5	11	27	43	59	75	91	107	123	B
4	12	28	44	60	76	92	108	124	C
3	13	29	45	61	77	93	109	125	D
2	14	30	46	62	78	94	110	126	E
1	15	31	47	63	79	95	111	127	F
0	16	32	48	64	80	96	112	·	·
	0	1	2	3	4	5	6	7	MSH·F

注释：1.计算从分支转移指令到分支转移目的指令的字节数

2.查找表中两边的数值

3.读下等效的十六进制数

a.向后分支转移时，为最上和最左一行数字

b.向前分支转移时，为最下和最右一行数字

例：向后15_{10}，字节=$F1_{16}$；向前77_{10}，字节=$4D_{16}$向后107_{10}，字节=95_{16}

4.缩写字含意：

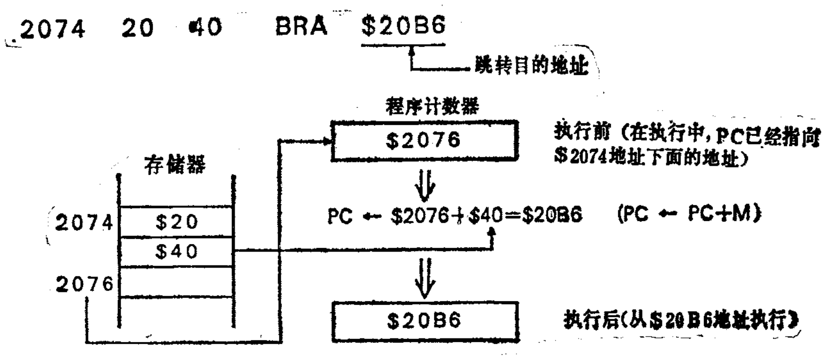
MSH-B=向后最高十六进数　　MSH-F=向前最高十六进数

LSH-B= 向后最低十六进数　　LSH-F=向后最低十六进数

$-32768_{10}\sim32767_{10}$，当整个程序位置独立时就可以实现向全部存储器空间的任何位置进行转移。这就是6809所用的长相对分支转移。

因为分支转移指令是控制程序流向的，所以在进行数据传送运算中是没有这种寻址方式的。

相对寻址方式之例见图3.13。



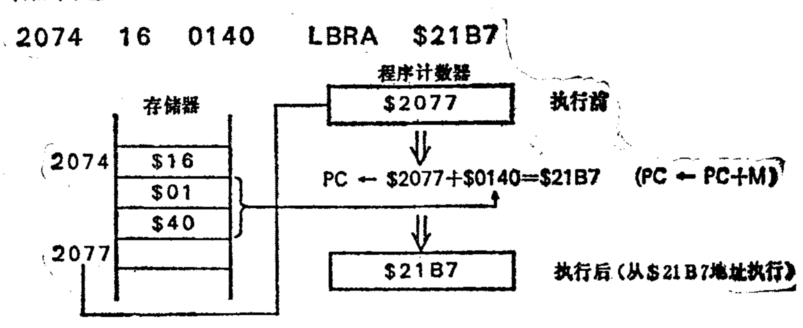


图 3.13 相对寻址方式之例

2. 程序计数器相对寻址

为了避免实现完全位置独立程序时的过多的软件开销，6809设有程序计数器相对寻址方式。使用这种寻址方式时，分配在存储器中操作数的地址可以由相对程序计数器的地址来决定。把相对PC寻址的 8 位或16位带符号的偏值加到程序计数器地址内容上，得到操作数的有效地址，或操作数地址的有效地址（后者属于间接寻址）。所以在6809的存储器空间的任何地址都可以相对程序计数器内容访问到任何数据，因此可以实现完全位置独立的程序。

程序计数器相对寻址的指令格式如图3.14所示。指令的长度可能是3、4或 5 字节，这取决于所要求偏值大小。允许有 8 位或16位偏值。两种偏值的后缀字节总要排在指令操作码之后。后缀字节之后为偏值字节。当用这种寻址时，程序计数器作为 8 位或16位偏值的指示寄存器；这样程序计数器相对寻址可以看成是一种变址寻址方式。所以凡是使用变址寻址方式的任何指令都可以使用程序计数器相对寻址。因为程序计数器寻址被作为一种变址寻址，所以需要用后缀字节把正在使用的程序计数器作为指示寄存器，而不是使用其它可变址的寄存器（X、Y、S、U）。

指令操作码（1或2字节）
后缀字节
±偏值

(a) 带符号的8位偏值

指令操作码　1或2字节
后缀字节
±偏值（高位字节）
偏值（低位字节）

(b) 带符号的16位偏值

图 3.14 程序计数器相对寻址

现以带符号的8位偏值采用计数器相对寻址为例说明如下：

```
.     .
.     .
.     .
00FF  .
0100  LDA（指令操作码）
0101  后缀字节
0102  10（8位偏值）
0103  .
.     .
.     .
.     .
```

该指令的意思是把带符号的偏值加到程序计数器内容后所决定的存储器地址单元中的内容装入到累加器A之中。带符号的偏值是10_{16}（0001 0000_2），后缀字节规定程序计数器即为指示寄存器。因为程序计数器总是要指出要执行的下一条指令，所以程序中在该点上程序计数器的内容为0103。这样要装入到累加器A的操作数被放在地址0103＋0010＝0113之中。在本例中偏值最高位为0，所以偏值为正数。如果偏值最高位为1，则偏值为2的补数是负。例如偏值为F0（11110000_2），有效操作数地址为0103＋FFF0＝00F3。

带符号16位偏值的程序计数器相对寻址之例如下：

```
.     .
.     .
00FF  .
0100  LDB（指令操作码）
0101  后缀字节
0102  01（偏值高字节）
0103  FF（偏值低字节）
0104  .
.     .
.     .
```

该指令的意思是把16位偏值（01FF）加到程序计数器内容之后所决定的存储器单元的

内容装入到累加器B之中。程序计数器在程序中该点的内容为0104。所以有效操作数地址为0104+01FF=0303。

3.2.7 变址寻址方式

6809的变址寻址方式在功能上比6800有很大的增强。它有四个16位的寄存器可在变址寻址方式中作为指示寄存器。这些寄存器为X、Y、S、U寄存器。6809变址寻址有四种基本形式都能用这些寄存器。它们是零偏值变址，常数偏值变址、累加器偏值变址和自动加／减变址。所以变址寻址方式是把指示器与偏值之和作为有效地址的寻址方式。除以上四个指示寄存器之外程序计数器也可以作为变址寄存器。

6809变址寻址的指令格式如图3.15所示。指令操作码之后总是为后缀字节，而偏值是可

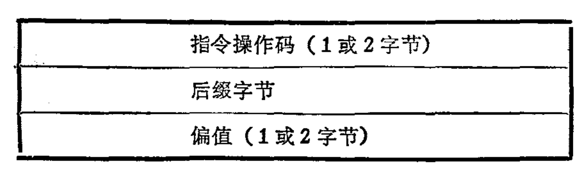


图3.15 一般变址寻址指令格式

有可无的。变址指令的后缀字节规定所用变址寻址的基本形式，以及确定有效操作数地址的指示寄存器。在6809的59条指令中有31条指令都可有变址寻址方式。关于后缀字节的含意和使用方法见表3.3的说明。

下面说明四种偏值方式的变址寻址：

（1）零偏值变址寻址

这种变址寻址可以用指示寄存器直接指出有效操作数地址，因为偏值此时为零。换句话说，被指定的指示寄存器本身包括操作中所用操作数的地址。这些指令可为2字节或3字节：即指令操作码字节①，后跟后缀字节。后缀字节将规定为零偏值方式和使用指示寄存器。这种变址寻址最快，因为所需字节数最少，而且也无需进行偏值计算。

（2）常数偏值变址寻址

这种变址寻址和6800的变址寻址很类似。但可以使用任何一个指示寄存器（X、Y、U、S）进行，而且带符号的偏值可以是5位、8位或16位的数值。在指令操作码之后为后缀字节，并由它规定指示寄存器和偏值范围。当使用带符号的5位偏值时，偏值为后缀字节的一部分，所以5位偏值在字节利用上最有效，同其它常数偏值型变址寻址相比，MPU周期数占用的最经济。偏值是2的补码（带符号的数值），即使用偏值的最高位来决定其符号。如果最高位为0，则偏值为正；如果最高位为1，则偏值为负。所以，5位的偏值为±4位偏值，其相应的偏值范围为-16_{10}～$+15_{10}$。

如果希望为8位偏值，指令长度将为3或4字节。这时指令操作码字节后跟后缀字节，其后再跟8位的偏值字节。后缀字节将规定常数偏值变址寻址和所用的指示寄存器。然后把偏值字节加到指示寄存器内容上，以便确定有效地址。再者，偏值为2的补数（带符号），其最高位决定符号。所以，8位偏值变为±7位偏值，其范围是-128_{10}～$+127_{10}$。

---

① 某些6809指令需要二个操作码字节

如果要求16位偏值，后缀字节后跟两个偏值字节。高位偏值字节在前，低位偏值字节在后。为了确定有效地址，需把带符号的16位偏值加到由后缀字节所决定的指示寄存器的内容中得到。因为偏值带符号，所以偏值变为±15位，其范围是-32768_{10}～$+32767_{10}$。

（3）累加器偏值变址寻址

这种变址寻址和常数偏值方式相似，区别只是把一个累加器的内容（ACCA、ACCB、或ACCD）加到指定的变址寄存器中（X、Y、S、U）来得到有效地址。这种方式的主要优点是在进行变址操作之前，偏值本身可以被计算出来。指令长度为2或3字节，指令操作码后跟后缀字节。后缀字节规定累加器偏值方式、指示寄存器和使用哪个累加器进行变址。6809用2的补数（带符号）作为累加器的内容来确定有效地址。至于指定哪个累加器或变址寄存器内容，这对偏值计算没有影响。

（4）自动加／减变址寻址

这种变址寻址是很有用的，因为在逐个单元查找存储器表格或传送存储器中数据块时，不需要单独使用变址寄存器内容加／减（1或2）的指令。在自动增加方式中，所规定的指示寄存器包括第一个操作数的地址。当按操作码指定的操作使用了第一个操作数之后，指示寄存器自动地增加给出下一个相邻的操作数的地址等等，可以多次地执行指令。所以依次从低到高的地址中取出存储器数据。在自动递减方式中，依次从高到低的地址取出存储器数据，因为在操作数被取出之前，在所指定的指示寄存器中其内容自动地减少。所以自动增加方式是后加操作；而自动递减方式是前减操作。因此，如果使用自动递减方式工作，为从n号地址中取出信息，起始地址必须是n＋1。加减的数量可以是1或者2，使用的数据可以是8位或者是16位。

自动加／减的指令长度为2或3字节，指令操作码后为后缀字节。后缀字节将规定自动增加还是自动减少，要增加或者减少指示寄存器、以及自动增加或减少的数量（1或2）。

变址寻址方式的具体操作原理和示例将在3.2.9节中说明，现只举变址寻址方式一例，如图3.16所示。后缀字节表示的含意请参阅表3.3的说明。

最后，为了全面清晰地说明变址寻址方式的概念，请见表3.7的有效变址方式组合表。

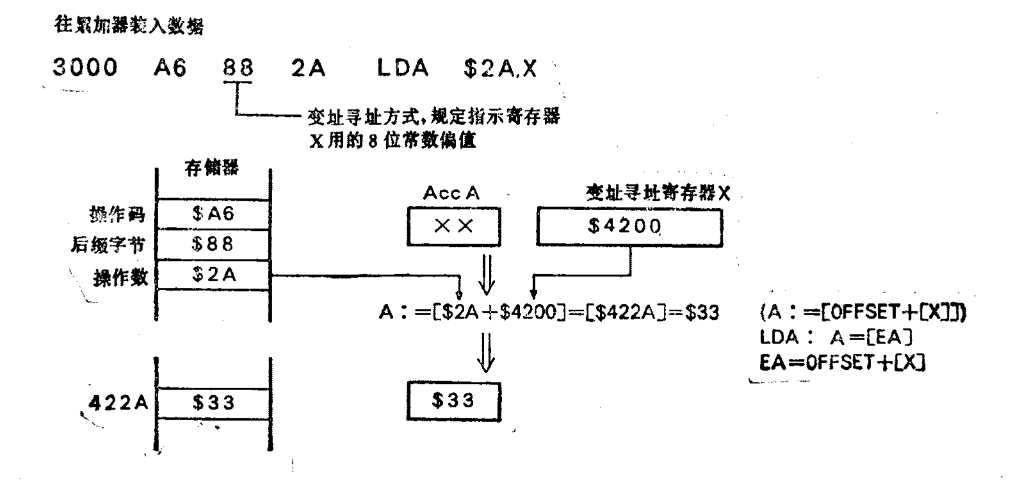


图3.16 变址寄存器寻址方式之例

表3.7 有效变址方式组合表

偏值 \ 操作数寄存器	X	Y	S	U	PCR	--X -X X+ X++	--Y -Y Y+ Y++	--S -S S+ S++	--U -U U+ U++
O									
常数 +或- 16位					注3				
A									
B									
D									

1.操作寄存器可以是A、B、D、X、Y、S、U

2.斜线区为无效区

3.当PCR为操作数寄存器时，汇编程序语句的偏值位置中的数值是存储器单元地址或标号，汇编程序计算偏值

3.2.8 间接变址寻址方式

6809的间接寻址方式的能力是很强的。间接寻址的方法是操作数的有效地址需由操作数来确定地址单元。得到操作数需要经由中间地址间接取得。

间接寻址是6809很重要的一种寻址方式，因为它可用在任何一种变址寻址方式之中（自动加／减除外）①。间接变址寻址方式是变址寄存器（扩充）和间接寻址方式相结合的寻址方式。把变址寻址方式的有效地址看作为间接寻址方式的操作数来确定有效地址值。这就是说，首先求出指示寄存器和偏值之和，这样就得到变址寻址方式的有效地址，而在间接变址寻址方式中，再把该有效地址作为操作数进行间接寻址。简言之，就是把指示寄存器与偏值之和所在存储器地址（2字节）的内容作为有效地址。

间接变址寻址方式之例如图3.17所示。例中的指示寄存器为X，偏值为8位常数。

各种变址寻址方式以及它们各自的汇编语言编码和后缀字节格式如表3.8所示。

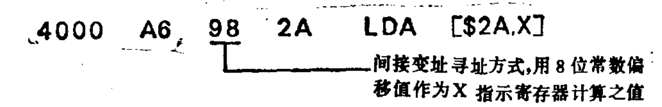
3.2.9 偏值的给定

1. 关于后缀字节

6809后缀字节的规定和作用在3.1.4节作过详细说明。现在从寻址方式的角度对后缀字节的应用加以总结。

现以LDA指令为例说明应用的情况。该指令的寻址方式可以有6种。即立即、直接、扩充、间接扩充、变址、间接变址等寻址方式。但其操作码＄86、＄96、＄A6、＄B6四种。因此使操作码＄A6对间接扩充、变址、间接变址三种寻址方式通用，而它们的区别只在于后缀字节的不同。如间接扩充寻址方式后缀字节＄9F，所有指令使用这种方式时后缀字节都是相同的，不需要其它任何选择。

① 间接寻址还可用在程序计数器寻址和扩充寻址之中。



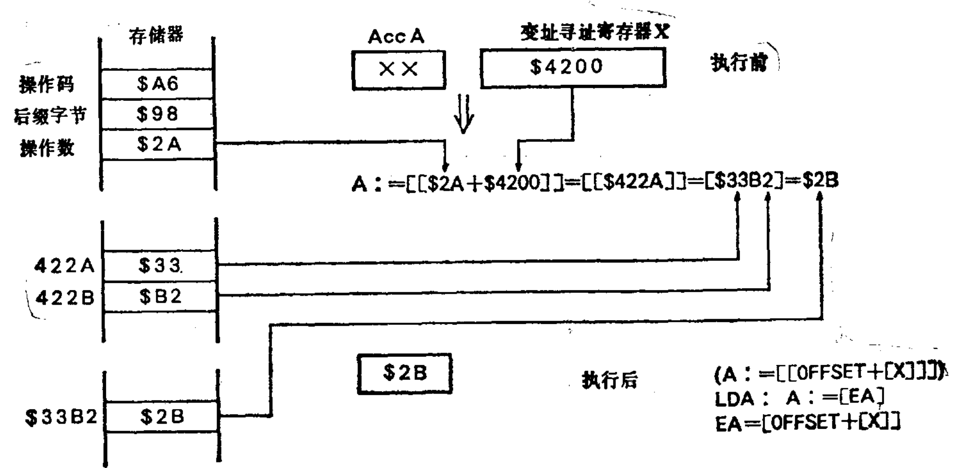


图 3.17 间接变址寻址方式之例

表3.8 变址寻址方式小结

种类	形式	直接				间接			
		汇编格式	后缀字节操作码	$\underset{\sim}{+}$	$\underset{\#}{+}$	汇编格式	后缀字节操作码	$\underset{\sim}{+}$	$\underset{\#}{+}$
用R作常数偏值（带符号偏值）	无偏值	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5位偏值	n,R	0RRnnnnn	1	0	无 8 位			
	8位偏值	n,R	1RR01000	1	1	[n,R]	1RR11000	4	1
	16位偏值	n,R	1RR01001	4	2	[n,R]	1RR11001	7	2
用R作累加器偏值（带符号偏值）	A寄存器偏值	A,R	1RR00110	1	0	[A,R]	1RR10110	4	0
	B寄存器偏值	B,R	1RR00101	1	0	[B,R]	1RR10101	4	0
	D寄存器偏值	D,R	1RR01011	4	0	[D,R]	1RR11011	7	0
R自动增/减	加1	,R+	1RR00000	2	0	不 允 许			
	加2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	减1	,−R	1RR00010	2	0	不 允 许			
	减2	,−−R	1RR00011	3	0	[,−−R]	1RR10011	6	0
用PC作常数偏值	8位偏值	n,PCR	1××01100	1	1	[n,PCR]	1××11100	4	1
	16位偏值	n,PCR	1××01101	5	2	[n,PCR]	1××11101	8	2
间接扩充	16位地址	—	—	—	—	[n]	10011111	5	2

R=X,Y,U或S　X=00　Y=01
×=任意　U=10　S=11
$\underset{\sim}{+}$和$\underset{\#}{+}$表示增加的周期数和特殊变化的字节数

另外从表3.8可知在变址和间接变址中，后缀字节总计可有23种。

所以对这三种寻址方式使用后缀字节即可决定三种偏值类型、二种自动选择和所有五个指示寄存器。

对于象$A6这样一种操作码的指令，用后缀字节来指定的内容有以下各项：

（1）寻址方式

间接扩充寻址方式

变址寻址方式

间接变址寻址方式

从表3.7可知，间接扩充寻址方式的后缀字节只有一种规定，即$9F，不能规定以下其它各项内容。而间接变址寻址方式中，以下各项内容有的也不允许规定，如5位偏值类型、和自动加／减1类型。

（2）偏值类型。

零偏值型

常数偏值型（对寄存器和程序计数器）

累加器偏值型

选择功能可以指定的只有零偏值类型；在指示寄存器中可指定程序计数器的只有常数偏值类型；在间接变址寻址方式中和常数偏值类型中，不能指定5位偏值类型。

（3）自动选择类型

自动增加（1或2）

自动减少（1或2）

只有零偏值类型才可指定。

（4）指示寄存器

对X、Y、U、S四个指示寄存器的指定与偏值类型、自动选择无关。程序计数器使用时，只能规定常数偏值。

2. 偏值的给定

（1）零偏值

零偏值含意是没有偏值，即把零作为偏值，没有操作数。因此在变址寻址方式中，有效地址就是指示寄存器中的内容。是唯一具有自动加和自动减功能的偏值类型。其操作原理和在常数偏值方式举例中，与使偏值为零时的情况相同。

应用零偏值可以指定的指示寄存器有X、Y、U、S四个寄存器。

（2）常数偏值

±4位偏值型（－16～＋15）

±7位偏值型（－128～＋127）

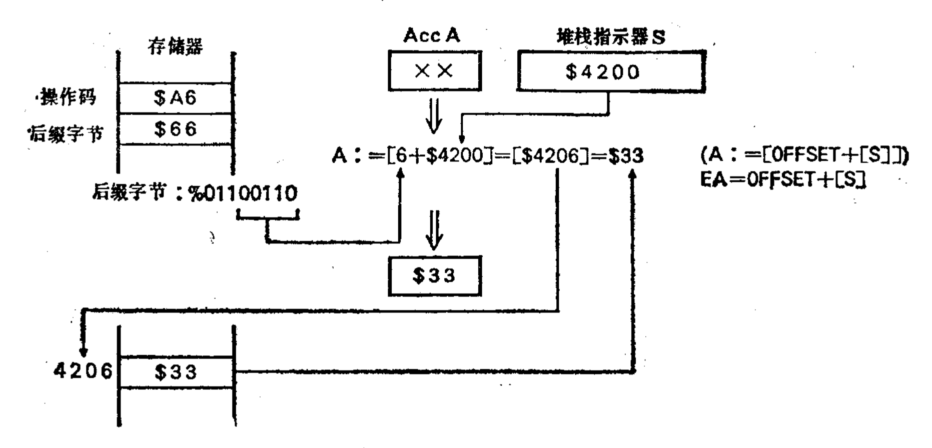
±15位偏值型（－32768～＋32767）

常数偏值根据后缀字节被分为三种类型。±4位偏值型的偏值数据包括在后缀字节之中。±7位偏值型的偏值数据为1字节。±15位偏值型偏值数据为2字节。它们都需设置1字节或2字节的操作数。

零偏值型和±4位偏值型的区别是：零偏值型中因为无需对偏值进行加法，所以利用一个机器周期即可迅速执行。

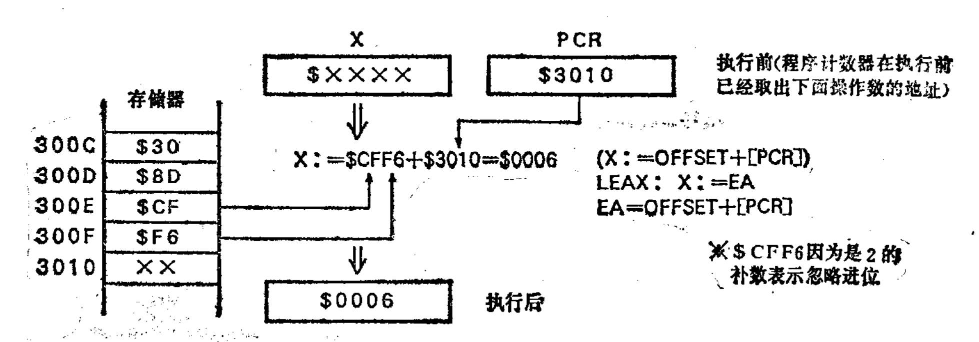
(a) 往累加器A装入数据（变址方式，±4位偏移值）

3002　A6　66　LDA　6, S



(b) 往X装入有效地址（±15偏值，变址寻址）

300C　30　8D　CFF6　LEAX　$CFF6, PCR



(c) 累加器A逻辑乘（间接变址±7位偏值）

4002　A4　D8　06　ANDA　[6, U]

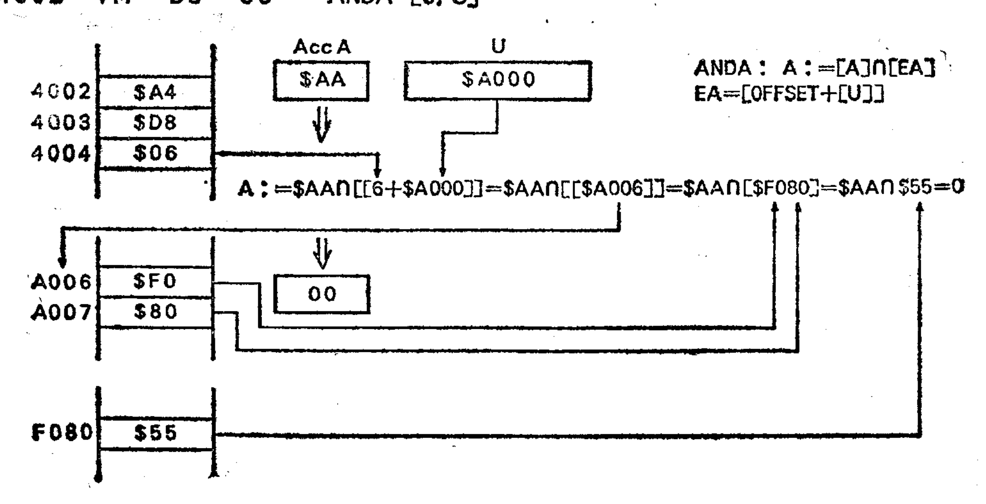


图 3.18　常数偏值之例

有关更详细的数字位的组合情况，请参阅表3.4或表3.8的有关后缀字节的内容。

使用汇编程序时，对各种类型的选择汇编程序可以自动地进行处理，程序设计人员使用n，R描述方法即可解决。根据汇编程序的规定，在存储器利用效率和速度上会有所不同。在R处可以使用X、Y、U、S和PCR。在±4位偏值型中不能使用PCR。

间接变址寻址方式中，没有±4位偏值的类型。常数偏值之例如图3.18所示，

（3）累加器偏值

在累加器偏值类型中，偏值由累加器A、B、或D给出。把累加器的内容按±7位（ACCA或ACCB）或16位（ACCD）加在指示寄存器（X、Y、U、S）的内容之中。

使用累加器类型的指示寄存器可以使用X、Y、U、S，不能使用程序计数器PC。

累加器偏值之例如图3.19所示。

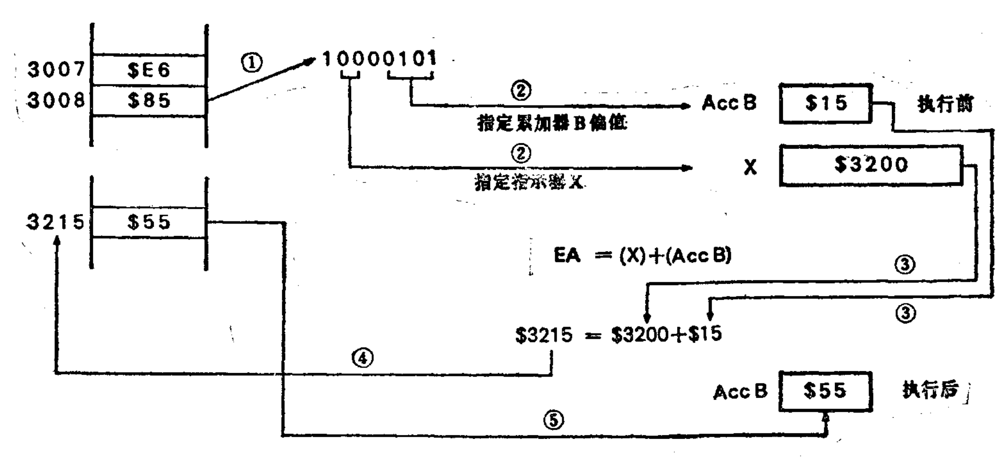


图3.19 累加器偏值之例

3. 自动选择

在零偏值类型中有自动增加和自动减少两种功能。

（1）自动增加

在自动增加方式中，由后缀字节所指定的指示寄存器（X、Y、U、S）的内容将作为有效地址输出到地址总线之上，当按基本指令所确定的处理结束之后，指示寄存器的内容则被加1或加2。因为自动增加在执行后进行加法，所以又称为“后增加”方式。使用汇编程序编程序时，表示这种后增加方式的描述方法，使用R＋或R＋＋的形式。

对于在表格内的步进或数据传送的操作，利用软件按自动增加方式做成堆栈工作形式是非常方便的。如在6800中（LDAA，X和INX）、6809中LDA，X＋这些程序即属这种情况。

在间接变址寻址方式中使用自动增加方式时需要考虑字节的数目。例如使用表格访问地址数据就是2字节的数据。所以在基于2字节的间接变址方式中只能使用自动加2（＋2）不能使用基于1字节数据的自动加1（＋1）。

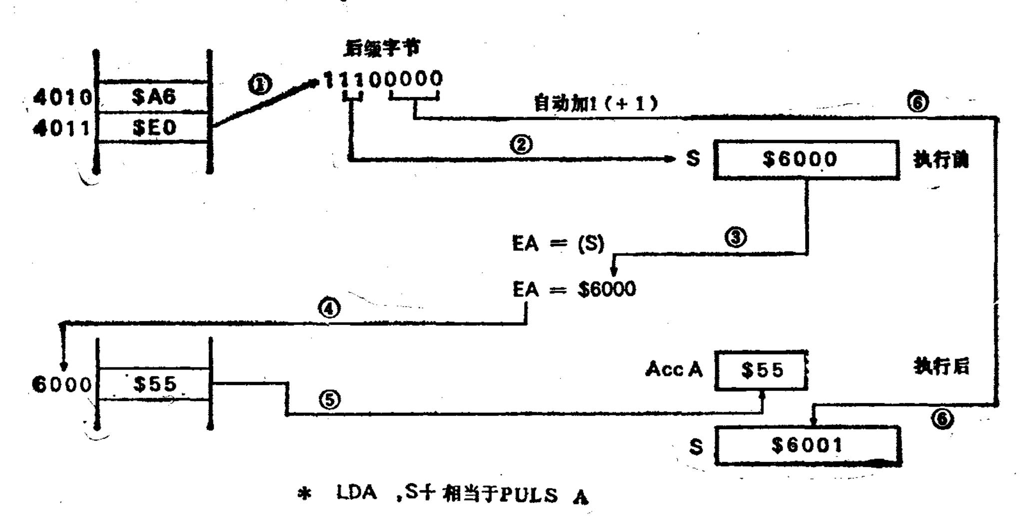
自动加2功能之例如图3.20所示。

（2）自动减少

自动减少功能是指：根据后缀字节所指定的指示寄存器，把其减1或者减2的内容作为

(a) 往累加器A装入数据（变址方式）

4010 A6 E0 LDA ,S+



(b) 往X装入数据（间接变址方式）

4020 AE B1 LDX [,Y++]

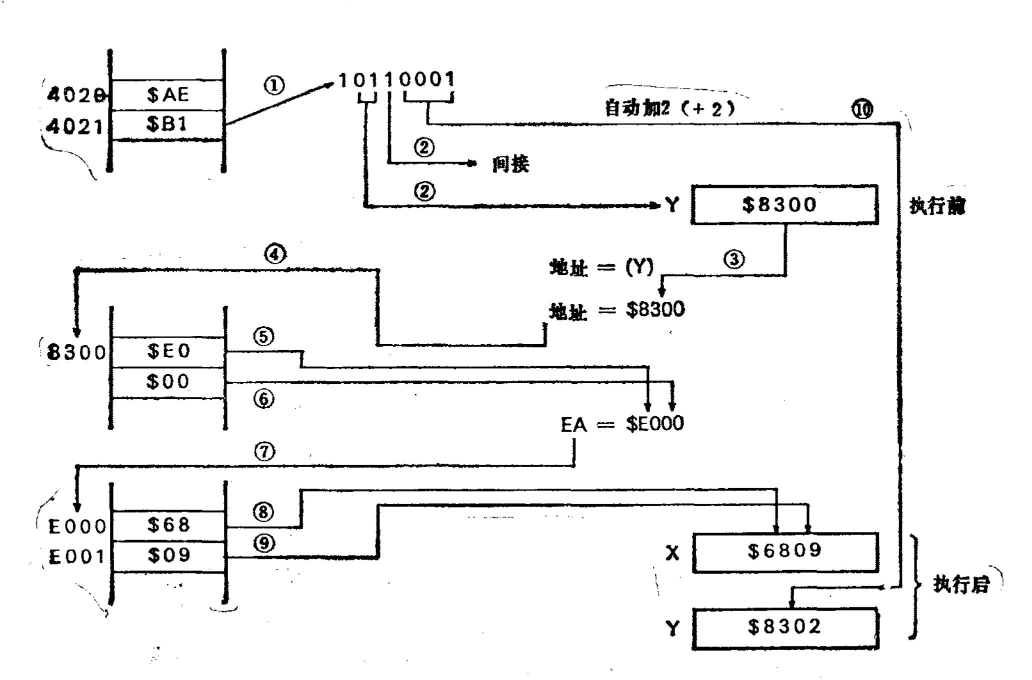


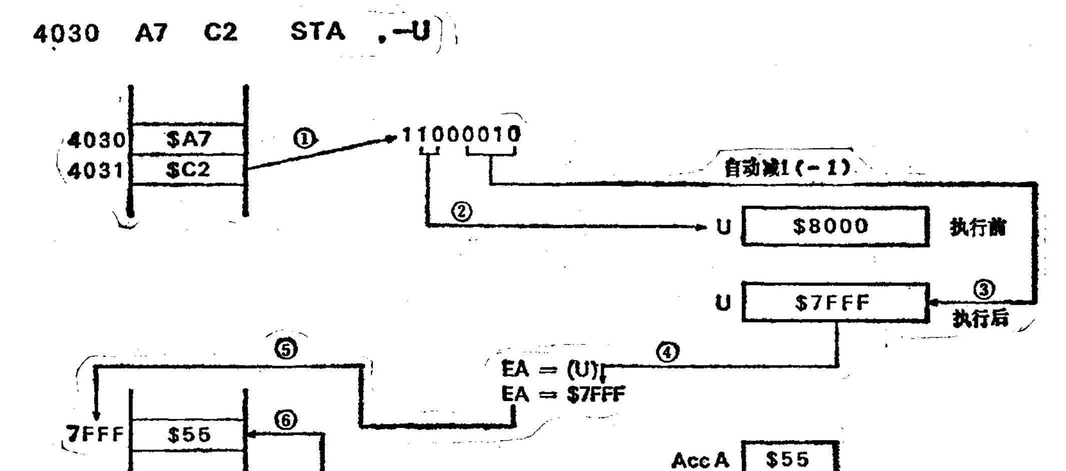
图 3.20 自动增加功能之例

有效地址（因为是零偏值型）。因此，在实际执行时，已经成为减1或者减2的内容。

因此，对指示器内容进行的减法操作要先行于基本指令的操作，所以称为“前减”。在汇编程序描述中和自动增加的情况不同，需在指定的寄存器之前规定是－1或－2。基本描

(a) 累加器A向存储器存储(变址方式)

4030 A7 C2 STA ,—U



(b) 累加器D的减法（变址方式）

4032 A3 A3 SUBD ,——Y

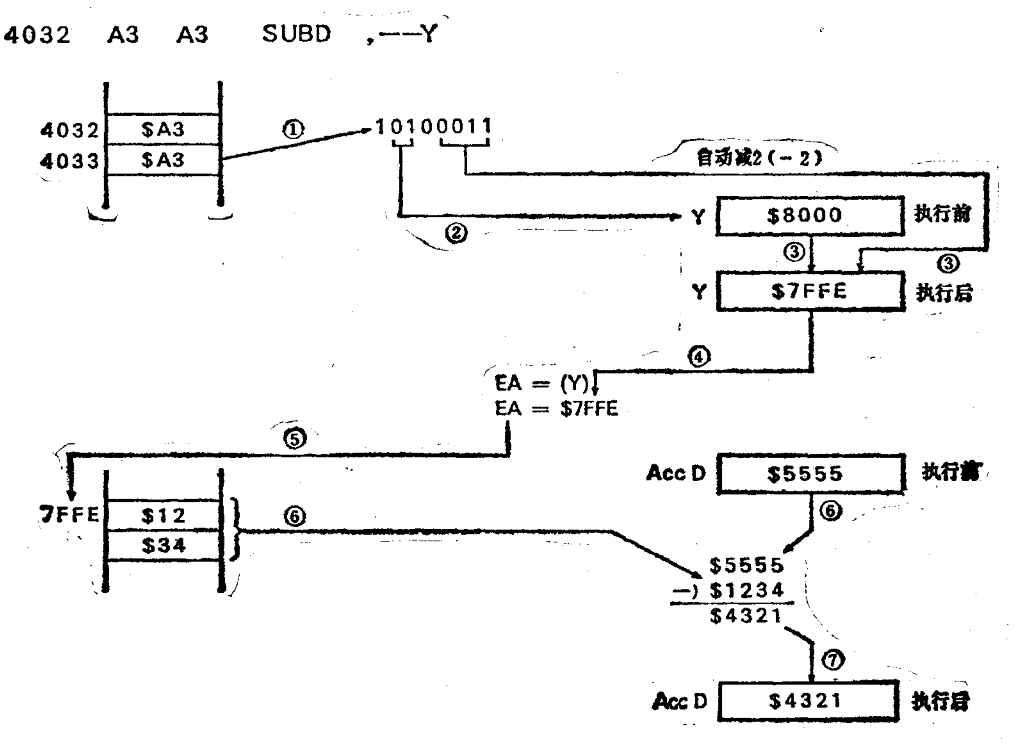


图3.21 自动减2功能之例

述方式是 ,－R或 ,－－R，用－表示的数相当于减数。

在间接寻址方式中的自动减少和自动增加一样，可以把有效地址数据从表格中顺序取出执行。因此利用这种方法时，不是访问一字节数据表格，只有－2的工作方式，－1的情况不存在。

自动减2功能之例如图3.21所示。

3.2.10 寻址方式小结

6809的寻址方式是按字节数的分配情况如表3.9(1)所示。寻址方式小结见表3.9(2)。

对寻址方式的概念性小结同寻址方式小结给出的内容。只要对本节所述内容有所了解，对这些概念性的说明应当一目了然。

表3.9(1)　6809寻址方式字节分配情况

字节	固有	立即	直接	扩充	相对	变址	间接扩充	间接变址
1 2 3 4	操作码	操作码 数据 8位	操作码* 地址L	操作码* 地址H 地址L	操作码 偏值	操作码* 后缀字节 无偏值	操作码* 后缀字节 地址H 地址L	操作码* 后缀字节 无偏值
1 2 3 4	操作码 后缀字节 TFR.EXG. PSH.PUL.	操作码* 数据H 数据L 16位			操作码* 偏值H 偏值L 长相对	操作码* 后缀字节 5位偏值		
1 2 3 4						操作码* 后缀字节 偏值 8位偏值		操作码* 后缀字节 偏值 8位偏值
1 2 3 4						操作码* 后缀字节 偏值 8位偏值-PC		操作码* 后缀字节 偏值 8位偏值-PC
1 2 3 4						操作码* 后缀字节 偏值H 偏值L 16位偏值		操作码* 后缀字节 偏值H 偏值L 16位偏值
1 2 3 4						操作码* 后缀字节 偏值H 偏值L 16位偏值-PC		操作码* 后缀字节 偏值H 偏值L 16位偏值-PC

续表

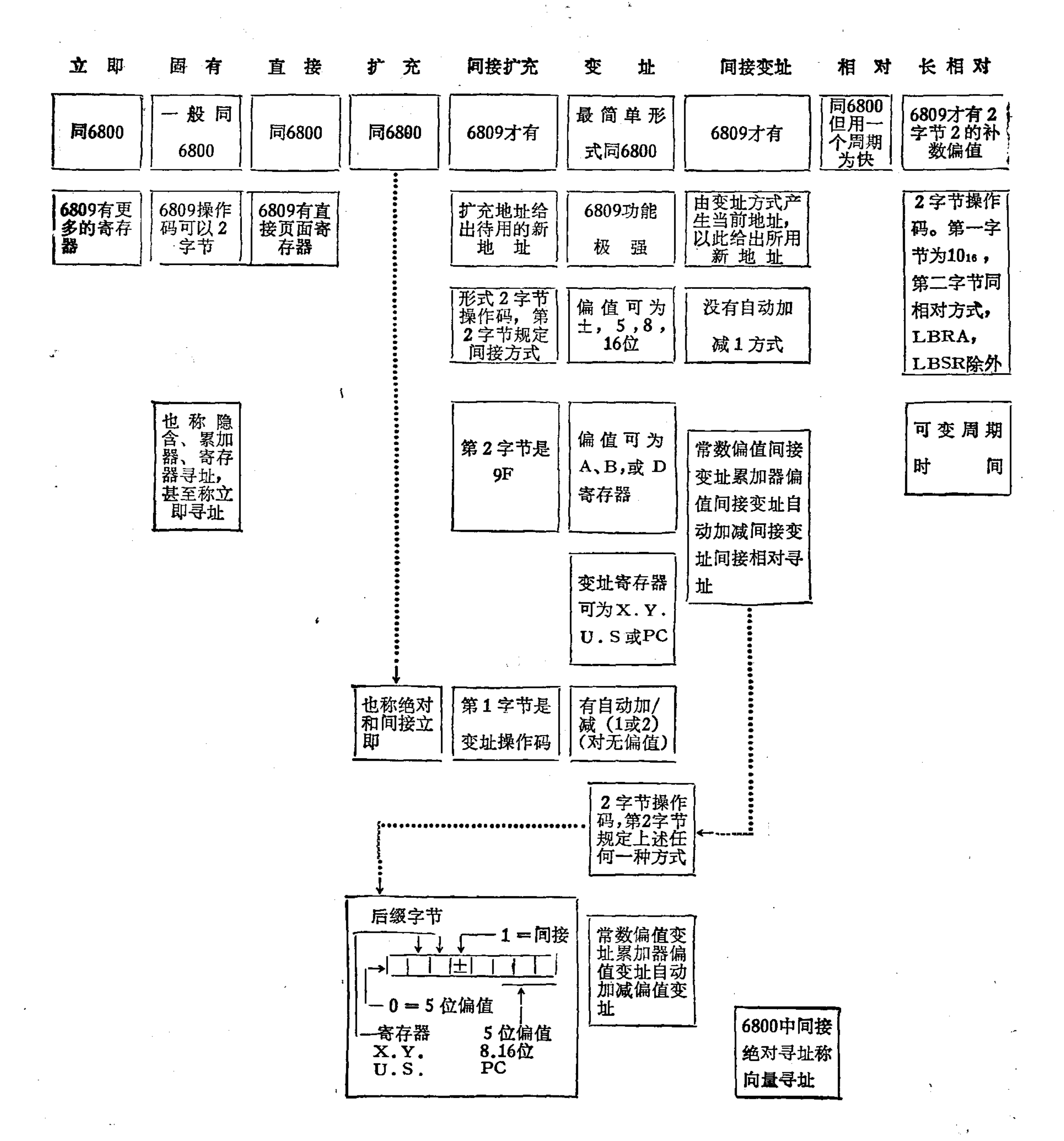
字节	固有	立即	直接	扩充	相对	变址	间接扩充	间接变址
1						操作码*		操作码*
2						后缀字节		后缀字节
3								
4						累加器偏值		累加器偏值
1						操作码*		操作码*
2						后缀字节		后缀字节
3								
4						自动加/减		自动加/减

*对某些指令将需要前置字节

无前置字节指令	前置字节为10	前置字节为11
SWI	SWI2	SWI3
SUBD	CMPD	CMPU
CMPX	CMPY	CMPS
LDX	LDY	—
STX	STY	—
LDU	LDS	—
STU	STS	—
分支转移	长分支转移*	

*对BRA,LBRA,BSR,LBSR例外

表3.9（2） 6809寻址方式小结



3.3 6809的指令分类

如前所述，6809的主要设计目标是在不牺牲同6800兼容的条件下，实现接近16位机性能的超级8位微处理器。所以6809在设计上扩充了6800的体系结构.同时对其指令系统作了认真的分析和清理，以便实现更有效处理任务。6800有72条基本指令，而6809的指令减少到59条。但

如果同各种寻址方式相组合，这59条基本指令即可以实现1464种基本操作，而6800只有197种基本操作。在规定6809指令系统之前，MOTOROLA公司对6800用户作了深入的调查，目的是确定6800有哪些指令最常用，而又有哪些指令不好用。调查分析的结果是最常用的为装入和存储，其次是子程序调用、转移、比较、增／减、清除以及加法／减法等。

在习惯上把6809的59条基本指令分为六类，它们是：数据移动、算术、逻辑、测试、转移和其它指令。以下将分别来说明这六类指令的基本功能和编程序的方法。

3.3.1 数据传送指令

6809的指令系统在数据移动指令方面作了重要的改进。例如数据传送和交换的操作利用TFR R_1，R_2和EXG R_1，R_2指令可以在任何两个长度相等的寄存器(R_1，R_2)中进行，用这两个记忆符可以形成63条基本指令，而不是有63个记忆符。这样就免去许多操作码，使用起来更方便有效。为取得和 6800 的兼容性，除新增加的指令外，6809 在源码上保持和 6800一致。但在目的码（操作码）级别上，为了更有效地进行指令编码，在设计中作了改动。

1.数据移动指令功能和操作

数据传送指令将完成以下操作：

（1）装入和存储寄存器A、B、D、X、Y、U和S；

（2）在任何两个长度相同的内部寄存器之间传送和交换数据；

（3）向X、Y、U和S寄存器中装入有效地址；

（4）在S或U的堆栈中，可以压入或弹出任何一个或者全部的内部寄存器的内容；

（5）在X、Y、U和S寄存器中加入 8 位数据；

（6）在X、Y、U和S寄存器中加入16位数据；

（7）在X、Y、U和S寄存器中加入A、B、D累加器内容；

（8）对X、Y、U和S寄存器加/减 1 或 2 。

实现上述操作的指令记忆符如表3.10所示。

由表3.10中可见，对任何一个累加器（A、B、D）或任何一个变址寄存器（X、Y、S、U）都可以进行装入或存储操作。装入操作可以用立即、直接、扩充、程序计数器相对或变址寻址方式实现；而存储操作可以用直接、扩充、程序计数器相对或变址等寻址方式实现。有关这些指令的操作码将在本节后面列表给出，当前我们最关心的是指令的意义。

从装入和存储操作符号中可以看到都含有16位寄存器，并需设有存储器 2 个字节或立即数。该 2 字节用M：M＋1符号来表示。指令本身将确定有效地址M和下个相邻存储器单元M＋1的地址。存储器地址M对应于被操作的寄存器的高位字节，地址M＋1对应于低位字节。在立即数寻址方式时，M和M＋1则分别为实际操作的高位和低位数据字节。例如一个 3 字节指令：

LDX $$

F1

C5

其含意是把存储器单元F1C5和F1C6中的内容装入X寄存器。这时M为F1C5，其内容将被装入到X寄存器的高位字节之中，而M＋1为F1C6，其内容将放到到X寄存器的低位字节中。另外下面的指令：

LDX •

F1

C5

其含意是把F1C5的本身装入X寄存器。

因为累加器D是累加器A和B的联合，所以装入累加器D的操作LDD实际上是高位字节装入累加器A、低位字节装入累加器B。另外，存储累加器D的操作STD实际是累加器A存到地址M中，累加器B的内容存到地址M+1之中。

3.10 数据移动指令

记	忆 符	操 作	操作符号
LD	LDA	立即数，或从存储器装入A	M→A
	LDB	〃 B	M→B
	LDD	〃 D	M:M+1→D
	LDS	〃 S	M:M+1→S
	LDU	〃 U	M:M+1→U
	LDX	〃 X	M:M+1→X
	LDY	〃 Y	M:M+1→Y
ST	STA	A的内容存到存储器	A→M
	STB	B 〃	B→M
	STD	D 〃	D→M:M+1
	STS	S 〃	S→M:M+1
	STU	U 〃	U→M:M+1
	STX	X 〃	X→M:M+1
	STY	Y 〃	Y→M:M+1
TFR	R1，R2	R1内容送到R2	R1→R2
EXG	R1，R2	R1内容和R2内容交换	R1→R2
LEA	LEAS	有效地址装入S	RA→S
	LEAU	〃 U	RA→U
	LEAX	〃 X	RA→X
	LEAY	〃 Y	RA→Y
PSH	PSHS	进入硬件堆栈(S)	无
	PSHU	进入用户堆栈(U)	无
PUL	PULS	出硬件堆栈(S)	无
	PULU	出用户堆栈(U)	无

传送（TFR）和交换（EXG）指令都是使用寄存器寻址的2字节指令，TFR或EXG的指令操作码之后为后缀字节，用它来指定在数据传送或交换中所用的寄存器（R_1和R_2）。该后缀字节的定义见图3.6的规定。各种寄存器的组合实际可有42种传送和交换操作，但它们都用这两条指令实现。只有一个要求是这两个寄存器的长度要相等。

在6809指令系统中，设有装入有效地址（LEA）的指令，表面看来这种优越的性能不是很明显。使用这些指令的目的是向所给出的指示寄存器（S、U、X、Y）装入按变址寻址方式有效地址（EA）进行计算所产生的数值。所以被装入指示寄存器的是有效地址数值而不是在有效地址中的数据。所有变址寻址方式，包括程序计数器相对寻址在内都可以使用LEA指令。

利用这些指令，在主程序中即可把数据字节的地址用变址寻址方式计算出来，然后进入

某一子程序，这种方法之所以可行是因为当调用子程序时指示寄存器的内容不被破坏。另外，这些指令还可以作为任何一个指示寄存器进行加或减法的操作的指令。这时可以加或减一个常数偏值或者任何一个累加器的内容。例如，LEAX－5，X含意是从X寄存器中减5；LEAS A，S含意是S寄存器的内容加上累加器内容；LEAY 10，U含意是U寄存器加上10并把其和传送到Y寄存器中。为使X寄存器加1可以使用LEAX 1，X实现；为使Y寄存器减1可以使用LEAY－1，Y完成。显然可以举出更多的例子。在汇编语言程序中经常可以看到各种变址寻址方式连接之中使用LEA指令进行。

6809的PSH和PUL指令可以使某个或任何数目的内部寄存器进出用户（U）存储器堆栈或者硬件（S）存储器堆栈。在讨论这些指令之前，首先说明有关6809的堆栈使用问题。首先，用户堆栈指示器（U）规定软件堆栈的起点，它不受子程序调用或中断应用的影响。所以用户堆栈完全由程序人员控制。相反，硬件堆栈指示器（S）规定了硬件堆栈的起点，为了在调用子程序和中断服务程序期间使内部寄存器数据进栈完全受6809自动地控制。在调用子程序期间，程序计数器自动地被保留在S堆栈之中，在中断处理期间，所有内部寄存器都被自动地保留在S堆栈之中。此时快速中断请求FIRQ除外，因为这时只有条件码寄存器和程序计数器被保留在堆栈之中。以前我们知道U和S堆栈指示器在变址寻址方式中可以作为变址寄存器使用；另外，它们还支持压入和弹出指令。这样6809就可以作为堆栈处理器使用，因此很容易支持象Pascal、FORTRAN等高级语言的使用。

所谓堆栈处理器是指该处理器可以使存储器无限地进行堆栈处理，这样极容易实现多级中断和子程序嵌套的处理。

在6800中堆栈指示器指向为存储器堆栈的上一个可用单元；但6809堆栈指示器实际所指向的是压入堆栈的最后一个单元数值，有时可称为栈顶。这种改变为的是使6809在自动增/减的变址寻址方式中，其X和Y寄存器还可作为软件堆栈指示器操作。例如STA，－X相当于以X寄存器作为地址规定的堆栈，用堆栈的指令使累加器A的内容进栈。再例如，指令LDA，X＋将把由X寄存器所规定的堆栈的顶部字节弹出到累加器A。这些指令的操作说明如图

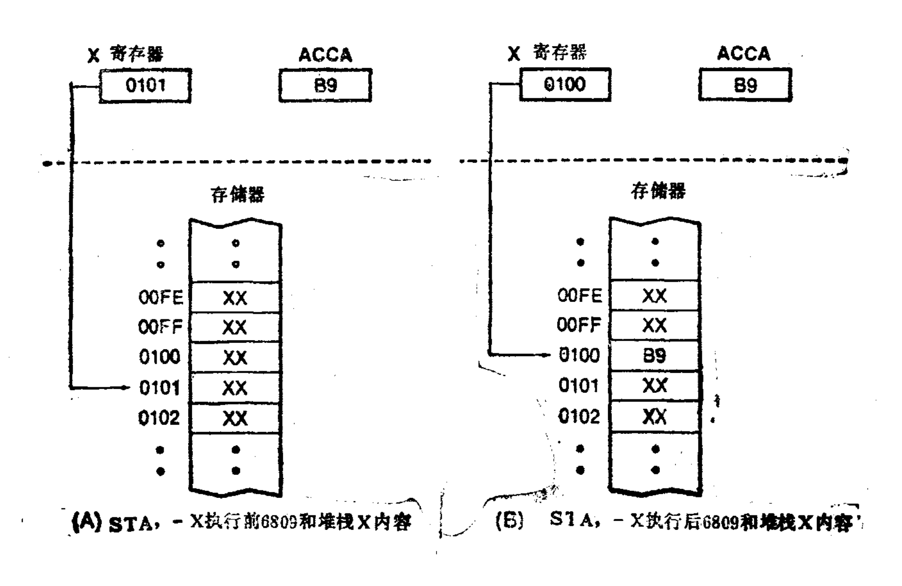


图3.22　STA，—X指令操作前后的状态

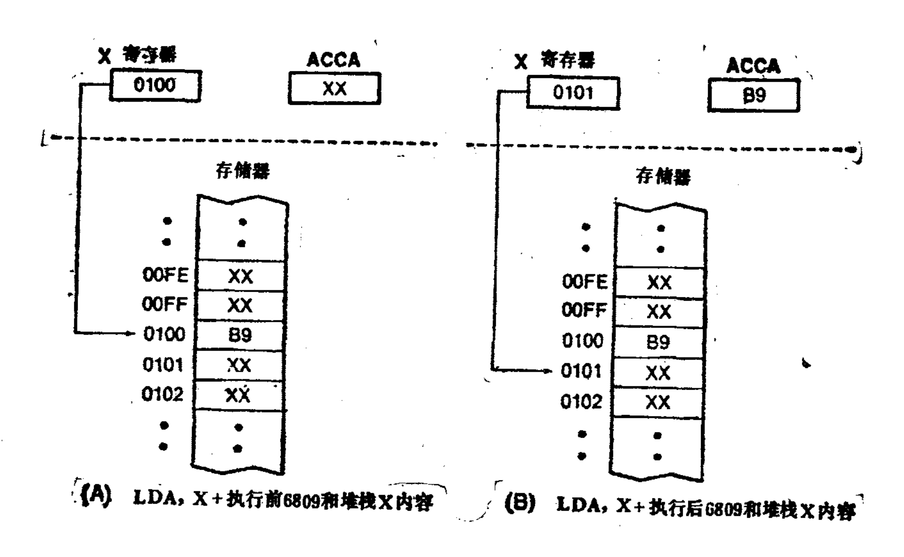


图3.23 LDA, X +指令操作前后的状态

3.22和图3.23所示。

可以用同种指令使累加器的数据进出堆栈，这时也可以使用其它的指示寄存器，不用上例所说的X寄存器，采用Y、S、或U均可。例如：STA, －Y；LDA,Y＋等。另外,仍然可以利用其它的存储（ST）和装入（LD）指令，以累加器B或D和寄存器X、Y、S、U分别作为操作的寄存器和指示器来进行进栈和出栈的工作，这些指令如表3.10所示有STB、STD、LDY、LDD等。但如果进出堆栈的是16位寄存器，那时就须使用自动加/减 2 的指令，因为堆栈中的数据需要两个字节。例如：STD, －－Y将把累加器D的内容压入Y寄存器确定的堆栈；而LDD, Y＋＋将把由Y寄存器确定的堆栈中顶部的两个字节弹出到累加器D。这些指令的操作说明如图3.24和图3.25所示。

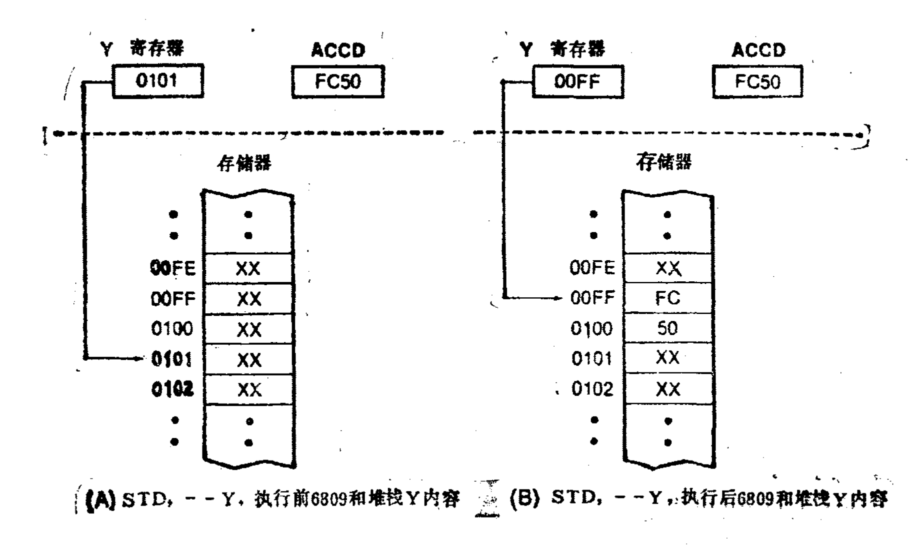


图3.24 STD, ——Y指令操作前后的状态

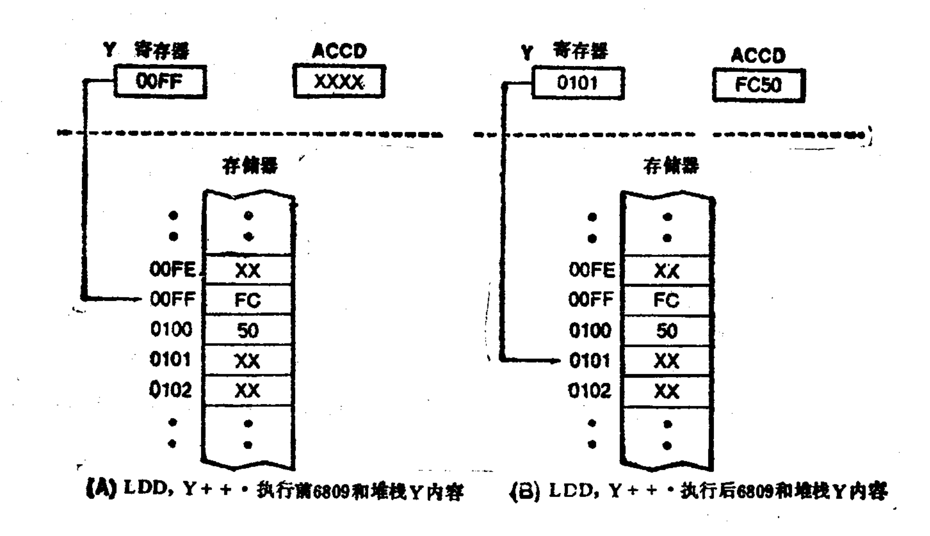


图3.25　LDD,Y ++指令操作前后的状态

既然可以软件设置堆栈，为什么还设置PSH和PUL指令呢？其原因在于增设这种指令的优点是可以使用S或U寄存器确定的堆栈，对6809的任何一个或者全部的内部寄存器进行进出堆栈的操作，此时只用一个2字节的指令。PSH和PUL指令需要两个字节：指令操作码和后缀字节。后缀字节要规定进栈或出栈的寄存器，后缀字节的格式如图3.26所示。数字位为1的表示将要进行堆栈操作的寄存器。寄存器进出堆栈的次序按图3.26标明的方向进行。例如要使累加器A、B和程序计数器PC进入U指定的堆栈时，其汇编程序代码为PSHU　A，B，PC。该指令所需后缀字节为10000110_2或86_{16}。

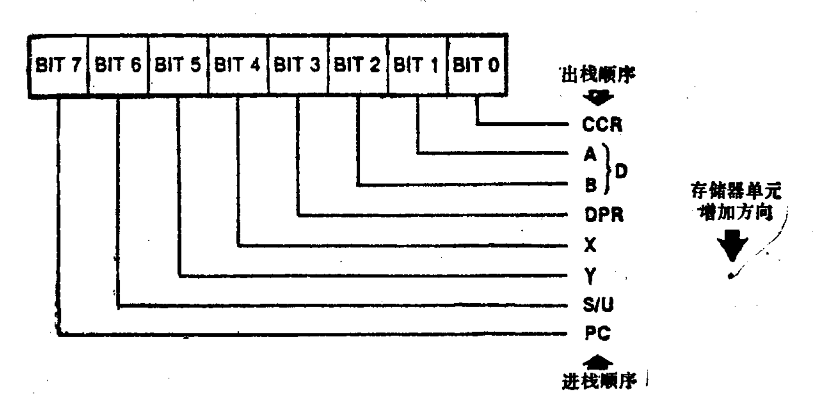


图3.26　PSH/PUL指令中后缀字节的格式

当使用PSHU或PULU指令时，后缀字节的第6位将对S寄存器进行堆栈操作；当使用PSHS或PULS时，后缀字节的第6位将对U寄存器进行堆栈操作。可以不需要所有寄存器进行堆栈操作，但局部进行堆栈操作时仍按图3.26所示的顺序。当进栈的是16位的寄存器时，低位字节先进栈，高位字节在后。16位的寄存器出栈时，高位在前、低位在后。如果如图3.26所示的寄存器全部进栈时，则需要堆栈存储器12个字节。还有，累加器D不规定堆栈操作，因为它实际上是累加器A和B的联合。

2. 数据移动指令操作码表

关于数据移动类型的6809指令操作码表如表3.11所示。该表中给出了每个基本指令在不同寻址方式时的操作码，同时也给出了相应的字节数和执行该指令时MPU所需要的时间周期数。另外在表中还给出了执行该指令后对条件码寄存器中各标志位的影响。在本表的下方写明了注意之点和表中所用符号的含意。从表中可知数据移动操作的基本指令有7条，基本操作码有73种。

3. 数据移动指令举例

（1）装入和存储直接页面寄存器

例： 写出用一个新数值装入直接页面寄存器，而把其原来数值存储起来的程序。

即使没有LDDP或STDP指令，用交换指令EXG实现该操作是相当简单的。如果设要装入直接页面寄存器DP的新数值为E5，原来DP中内容存入存储器E510单元。此时可以编出如下指令的程序：

```
LDA #
E5
EXG A,DP
STA $
10
```

以上指令操作的顺序是：用立即数寻址方式把数据字节E5装入累加器A，然后累加器A和直接页面存储器中的内容进行交换，这时新的数值E5在DP寄存器中，而DP内容在累加器A中。因为要求把原来DP内容存到存储器单元E510，而DP的新内容即为E5，所以可用直接寻址方式完成存储操作。这样使用直接寻址把累加器A的内容（原来DP寄存器内容）存到存储器单元E510，此时DP寄存器内容为E5作为高位地址字节，指令中直接地址10作为低位地址字节。

在该程序中EXG指令所需要的后缀字节应为10001011_2或$8B_{16}$（见图3.6）

同时从表3.11中可知，该程序需要占用存储器6个字节，执行时用MPU的13个机器周期。

（2）把数据加到指示寄存器中

例： 写出把累加器A的内容加到Y寄存器，然后把其结果送到S寄存器的程序。

6809没有ADD指令，所以为了把数据加到任何一个指示寄存器，可以使用装入有效地址LEA指令达此目的。指令LEAS A,Y即可完成上述要求。

首先使累加器A的内容加到Y寄存器的内容形成有效地址（EA），然后用LEAS指令使有效地址送到S寄存器。

上述指令所要求的后缀字节为10100110_2或$A6_{16}$（见表3.4）。

另外用下面的指令也可以实现同样的任务。

```
LEAY A,Y
TFR  Y,S
```

从表3.11可知LEAS A,Y指令需要2+字节和4+ MPU周期。这就是说，为了确定准确

表 3.11 数据移动指令操作码表

指令形式	6809寻址方式																		说明	5	3	2	1	0
	固有			直接			扩充			立即			变址			相对								
	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~[5]	#		H	N	Z	V	C
EXG R1,R2	1E	7	2																R1↔R2[2]	•	•	•	•	•
LD LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M→A	•	↕	↕	0	•
LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M→B	•	↕	↕	0	•
LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M:M+1→D	•	↕	↕	0	•
LDS				10 DE	6	3	10 FE	7	4	10 CE	4	4	10 EE	6+	3+				M:M+1→S	•	↕	↕	0	•
LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M:M+1→U	•	↕	↕	0	•
LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M:M+1→X	•	↕	↕	0	•
LDY				10 9E	6	3	10 BE	7	4	10 8E	4	4	10 AE	6+	3+				M:M+1→Y	•	↕	↕	0	•
LEA LEAS													32	4+	2+				EA[3]→S	•	•	•	•	•
LEAU													33	4+	2+				EA[3]→U	•	•	•	•	•
LEAX													30	4+	2+				EA[3]→X	•	•	↕	•	•
LEAY													31	4+	2+				EA[3]→Y	•	•	↕	•	•
PSH PSHS	34	5+[4]	2																进S栈	•	•	•	•	•
PSHU	36	5+[4]	2																进U栈	•	•	•	•	•
PUL PULS	35	5+[4]	2																出S栈	•	•	•	•	•
PULU	37	5+[4]	2																出U栈	•	•	•	•	•
ST STA				97	4	2	B7	5	3				A7	4+	2+				A→M	•	↕	↕	0	•
STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	•	↕	↕	0	•
STD				DD	5	2	FD	6	3				ED	5+	2+				D→M:M+1	•	↕	↕	0	•
STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S→M:M+1	•	↕	↕	0	•
STU				DF	5	2	FF	6	3				EF	5+	2+				U→M:M+1	•	↕	↕	0	•
STX				9F	5	2	BF	6	3				AF	5+	2+				X→M:M+1	•	↕	↕	0	•
STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y→M:M+1	•	↕	↕	0	•
TFR R1,R2	1F	7	2																R1→R2[2]	•	•	•	•	•

注解:

1. 表中所给为基本周期和字节数。确定总周期和字节数需加上6809变址方式表中所给的数值。
2. R1和R2可以是任何8位寄存器对或任何16位寄存器对。
   8位寄存器是：A、B、CC、DP
   16位寄存器是：X、Y、U、S、D、PC
3. EA是有效地址
4. PSH和PUL指令本身需要5个周期，凡是每进出栈一个字节，再加一个周期。
5. 5(6)含意是：不转移时5周期，转移时6周期。
6. SWI 1位和F位置1，SWI2和SWI3不影响I和F位。
7. 条件码置位情况为指令执行的直接结果。
8. 半进位位的标志位数值不受影响。
9. 特殊情况——如b_7为1时，进位位置1。

符号:

OP 操作码（十六进制）；
~ MPU周期数；
程序字节数；
+ 算术加；
− 算术减；
* 乘法；
$\overline{M}$ M的反码；
→ 传送方向；

H 第3位的半进位位；
N 负数位（符号位）；
Z 零位（字节）；
V 溢出位（2的补码）；
C 第7位的进位位；
↕ 测试，真时置1，否则置0；
· 不受影响；
CC 条件码寄存器，

: 联接符；
∨ 逻辑或；
∧ 逻辑与；
⊻ 逻辑异或。

的字节数和MPU周期，需要根据所给的操作，加上6809变址寻址方式表中（表3.4）的数值。从表3.4可知该操作不需要增加字节数，而只增加一个MPU周期。所以LEAS A，Y指令需要2字节，执行时需要5个周期。同理，第二种程序需要4个存储器字节，执行时需要12个MPU周期。虽然这两种程序的字节数和周期数不同，但完成的任务相同。

（3）堆栈操作

例：　写出根据Y寄存器所规定的堆栈，使累加器A、DP寄存器和U堆栈指示器进入堆栈的程序。

当以Y寄存器作堆栈指示器时，进出堆栈的操作需要根据寄存器的长度，使用自动加/减1或2的工作方式。以下指令可完成所给任务：

```
TFR  DP, B
STD, - - Y
STU, - - Y
```

因为DP寄存器不能被直接存储，所以先用传送指令使DP寄存器内容传送到累加器B之中，根据要求需把累加器A和B存储到存储器堆栈之中，因此可使用STD指令实现。STD，－－Y，指令依次使累加器B和A先后进入Y指定的堆栈。然后STU，－－Y指令使U寄存器进入Y的堆栈，U的低位字节在前，高位字节在后。在执行完指令之后，Y寄存器指示的将为堆栈的顶部单元，也就是含有U寄存器高位字节的存储器单元。而累加器B放在堆栈底部。

因为源目寄存器分别为直接页面寄存器和累加器B，所以传送指令TFR的后缀字节为10111001_2或为$B9_{16}$（见图3.6）。STD　，－－Y指令和STU　，－－Y指令的后缀字节都是10100011_2或$A3_{16}$，因为它们的寻址方式都是变址寻址，而且使用的是同一个指示寄存器(见表3.4）。

现在如要求写出以上寄存器出栈的指令程序时，其内容如下：

```
LDU  , Y + +
LDD  , Y + +
TFR  B,DP
```

这时出栈的次序与原来它们进栈的次序正好相反，因为堆栈寄存器使用的是后进先出(LIFO）原理实现的。上述指令的后缀字节是$A1_{16}$、$A1_{16}$和$9B_{16}$（见图3.6、表3.4）。

（4）堆栈指令的操作

例：　写出使累加器A、CCR、DPR、PC和X寄存器进入用户堆栈的程序。

实现该任务只需要一条2字节的指令：PSHU A,CC,DP,PC,X

该指令的第1字节是指令操作码，从表3.11可知为36_{16}，第2字节为后缀字节，从图3.26可以确定需要使所给寄存器数据进栈的后缀字节是10011011_2或$9B_{16}$。

根据图3.26所示进栈次序是：PC、X、DPR、A、CCR。其中CCR在栈顶（最低的存器地址），PC在栈底（最高的存储器地址）。

（5）其它数据传送操作

最后利用本节介绍的6809指令实现数据传送操作的举例如下：

```
交换A和X          = PSHS  A,B
                    TFR   X,D
  (X_H→A)           PULS  A
```

```
                          TFR  D,X
  (A:X_L→X)               PULS B

交换B和X                 =PSHS A
                          PSHS B
  (X_L→B)                 TFR  X,D
                          PULS B
                          TFR  D,X
  (X_H:B→X)               PULS A

传送A到X                 =PSHS X
                          STA  0,S
  (A:X_L→X)               PULS X

传送B到X                 =PSHS X
                          STB  1,S
  (X_H:B→X)               PULS X
```

3.3.2 算术、逻辑和测试指令

6809的大部分算术、逻辑和测试指令，除去它们所用的寻址方式之外都和6800非常相似。但也有些例外。如算术和测试指令可以使用累加器D进行，其中设有最有用的一种指令就是乘法MUL指令，该指令可以对累加器A和B中的内容实行乘法，16位的乘积存在累加器D之中。另外还有使累加器B的符号位扩充到累加器D的指令，如8位带符号的数值(2的补数)可以被扩充到16位带符号的数值（2的补数）。

6809的逻辑指令可以使在累加器A、B或存储器中的内容实现与、或、异或、变反、移位等操作。为了置位和清零条件码寄存器的标志位，6809没有单独设置固有的指令，而是使用两条逻辑指令：ANDCC和ORCC实现。这两条6809的指令代替了6条6800指令，所以空出了操作码的数值，可用在其它更有效的地方。

1. 算术指令

（1）算术指令的功能和操作

6809的算术指令可以实现以下操作：

- 在A、B、D中加或减存储器的内容；
- 在A、B、D中加或减带进位的存储器内容；
- A、B、D和存储器中内容增/减；
- 清除A、B和存储器单元；
- A、B和存储器内容变反；
- 按二—十进制数（BCD）运算；
- A和B进行乘法；
- B的符号扩充到D。

6809的算术指令如表3.12所示。这些指令有许多和6800类似，但6809有更多的寻址方式。这里不详细介绍每条指令，而是重点说明三条新的指令：ABX（B加到X），MUL（乘法），和SEX（带符号B扩展到A）。

ABX指令是把累加器B的无符号的8位二进制数加到X寄存器的内容上，其结果放在X寄存器中。ABX是1字节的固有寻址指令，该指令很类似于LEAX B,X。但ABX指令是

表 3.12　6809算术指令

记忆符		操　作	操作符号
ABX		B加到X（无符号）	B+X→X
ADC	ADCA	存储器加到A，带进位	A+M+C→A
	ADCB	存储器加到B，带进位	B+M+C→B
ADD	ADDA	存储器加到A	A+M→A
	ADDB	〃　B	B+M→B
	ADDD	〃　D	D+M:M+1→D
CLR	CLRA	清零A	0→A
	CLRB	清零B	0→B
	CLR	清零存储器	0→M
DAA		十进制调整A累加器	无
DEC	DECA	累加器A减1	A-1→A
	DECB	〃　B减1	B-1→B
	DEC	存储器减1	M-1→M
INC	INCA	累加器A加1	A+1→A
	INCB	累加器B加1	B+1→B
	INC	存储器加1	M+1→M
MUL		A乘B（无符号）	A×B→D
NEG	NEGA	累加器A变补（2的补数）	$\overline{A}$+1→A
	NEGB	累加器B变补（2的补数）	$\overline{B}$+1→B
	NEG	存储器变补（2的补数）	$\overline{M}$+1→M
SBC	SBCA	累加器A减存储器（带借位）	A-M-C→A
	SBCB	累加器B减存储器（带借位）	B-M-C→B
SEX		累加器B符号扩充到A	无
SUB	SUBA	累加器A减存储器	A-M→A
	SUBB	累加器B减存储器	B-M→B
	SUBD	累加器D减存储器	D-M:M+1→D

把累加器B作为0～255之间的无符号的正偏值。而LEAX B,X是把累加器B作为在-128～+127之间带符号的2的补数偏值。另外，LEAX B,X指令需要2字节，而ABX为1字节，所以当累加器B中存在着比较大的正偏值时，采用ABX指令有利。

在高级语言中，经常需要对各种信息矩阵进行计算。这些计算中经常要用到乘法，所以在6809指令系统中增设了乘法指令（MUL）。乘法指令本身可以在6809中直接进行乘法，而无须使用专门的算法。该MUL指令是使累加器A和B中的无符号的（8位二进制）数值内容直接在一起相乘，而其所产生的无符号的16位的结果被放在累加器D之中。指令内部的执行过程如图3.27所示。从该图中可知，累加器A和B的内容在算术逻辑部件（ALU）中相乘，其结果被放在累加器D中。因为累加器A和B共同组成累加器D，所

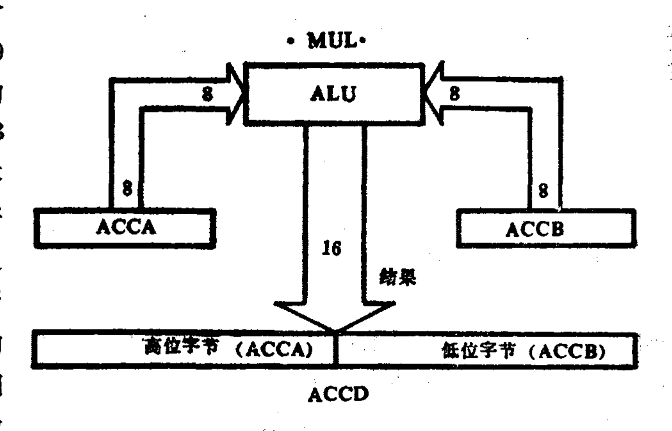


图3.27　乘法MUL指令执行过程

以结果的高位字节放在累加器 A，低位结果放在累加器B。因此这两个累加器中相乘前的内容（乘数和被乘数）就被丢失。MUL 指令是一种一字节的固有寻址指令。为了简化多精度的乘法（多字节），6809的乘法采用了无符号的乘法，而带符号的乘法不适应这种情况。

最后，6809设有SEX作为符号扩充指令。该指令的作用是可把累加器 B中的带符号 8 位数值（2 的补码）转换为累加器D中的一个16位的 2 的补码数值。实际上，SEX 指令是使累加器B 的最高位（符号位）移动到累加器 A的最高位。所以，这就很容易把一个 8 位的带符号的数为后来的内部16位操作而转换为一个16位的数。例如，外部采用 8 位字长作数据通信用，但内部，6809可进行16位操作，所以需要这种转换。SEX指令也是一种 1 字节的固有指令。

（2）算术指令操作码表

算术指令的操作码表如表3.13所示。该表中的有关符号说明见表3.11中的注解。

（3）算术指令操作举例

乘法指令的应用

例： 说明下述指令程序的任务：

```
LDA  #
  20
LDB  #
  0A
MUL
STD  [,X]
```

该指令程序首先在累加器A中装入20，然后在累加器 B中装入 0A。再用乘法指令，两数相乘，其结果为0140放在累加器D中。因为累加器 D 是A和B的组合，所以累加器A中内容应为01，累加器 B中为40。而以前的累加器中内容丢弃。最后，其结果使用 STD 指令按X指示寄存器（无偏值）作间接寻址进行存储。所以，X 寄存器的内容所规定的地址是：高位结果字节存储单元的高位地址字节；而高位结果字节存储单元的低位地址字节为变址寄存器内容加 1，即X + 1 所规定的地址。例如，设变址寄存器内容为0100，此时X = 0100，X + 1 = 0101。还设存储单元0100内容为FC，0101内容为50；此时，高位结果字节被存在 FC50单元，低位结果字节被存在FC51单元。

根据表3.4、表3.11和表3.13可知以上程序的操作码为：

```
LDA #       86
  20        20
LDB #       C6
  0A        0A
MUL         3D
STD [,X]    ED
            94
```

从程序的操作码和数据可知需要 7 个字节。其执行时需要 9 个MPU周期。

计算堆栈指示器

例： 写出计算X寄存器偏值的指令程序，其要求是：把按Y寄存器所规定的两个相邻的存储器单元的内容相加，当偏值计算完成以后把它加到X寄存器，以便设立X堆栈指示器。然后再把S和U寄存器保留在X堆栈之中。

表3.13 算术指令操作码表

指令形式		6809寻址方式 固有 OP	固有 ~	固有 #	直接 OP	直接 ~	直接 #	扩充 OP	扩充 ~	扩充 #	立即 OP	立即 ~	立即 #	变址 OP	变址 ~	变址 #	相对 OP	相对 ~	相对 #	说明	5 H	3 N	2 Z	1 V	0 C
ABX		3A	3	1																B+X→X 无符号	•	•	•	•	•
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A+M+C→A	↕	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B+M+C→B	↕	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A+M→A	↕	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B+M→B	↕	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D+M:M+1→D	↕	↕	↕	↕	↕
CLR	CLRA	4F	2	1																0→A	•	0	1	0	0
	CLRB	5F	2	1																0→B	•	0	1	0	0
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0→M	•	0	1	0	0
DAA		19	2	1																10进制调整A	•	↕	↕	0	↕
DEC	DECA	4A	2	1																A−1→A	•	↕	↕	↕	•
	DECB	5A	2	1																B−1→B	•	↕	↕	↕	•
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M−1→M	•	↕	↕	↕	•
INC	INCA	4C	2	1																A+1→A	•	↕	↕	↕	•
	INCB	5C	2	1																B+1→B	•	↕	↕	↕	•
	INC				0C	6	2	7C	7	3				6C	6+	2+				M+1→M	•	↕	↕	↕	•
MUL		3D	11	1																A×B→D 无符号	•	•	↕	•	9
NEG	NEGA	40	2	1																$\overline{A}$+1→A	8	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$+1→B	8	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$+1→M	8	↕	↕	↕	↕
SEX		1D	2	1																B中符号扩入A	•	↕	↕	0	•
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A−M−C→A	8	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B−M−C→B	8	↕	↕	↕	↕
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A−M→A	8	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B−M→B	8	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D−M:M+1→D	•	↕	↕	↕	↕

其程序如下：

```
 LDB       , Y
ADDB     1, Y
 ABX
 STS    , --X
 STU    , --X
```

LDB指令是使累加器B装入由Y寄存器所规定的存储器单元的内容，因为此时为零偏值。然后，用ADDB指令把相邻的下一个存储器单元内容加到累加器B，其结果在累加器B中（由Y+1作下个存储器单元，因为偏值为常数1）。计算偏值时，用ABX指令把累加器B中计算的偏值加到X寄存器，其结果放在X寄存器之中，以完成设置X堆栈指示器的任务。然后用STS和STU指令，使X寄存器自动减2，以便把S和U寄存器保留在X堆栈之中。

根据表3.4、3.11和3.13可知，其程序指令的操作码是：

```
 LDB     , Y        E6
                    A4
ADDB  1 , Y         EB
                    21
 ABX                3A
STS     , --X       10
                    EF
                    83
STU     , --X       EF
                    83
```

该程序中的的ADDB指令中的常数偏值1为后缀字节的一部分；STS指令为2字节的操作码。由上述各表中可和执行时需要29个MPU周期。

2. **逻辑指令**

（1）逻辑指令的功能和操作

6809的逻辑指令可以实现以下操作：

• 累加器A或B同存储器单元内容进行与、或、异或操作，结果在累加器A或B中。

• 存储器、累加器A或B进行移位和循环移位。

• 存储器、累加器A或B中内容变为反码。

• 条件码寄存器内容同立即数相逻辑与，或逻辑或，结果放在CCR中。

6809逻辑指令如表3.14所示。和6800系列一样设置了标准的逻辑操作与、或及异或，同时还设有各种算术和逻辑的移位操作。对这些指令的详细讨论请参考“微型计算机基础技术手册”——科学出版社出版的6800专门书籍。从表3.14中可知，算术左移（ALS）和逻辑左移（LSL）都用相同的操作符号，这是因为它们实现的操作完全相同，而且它们的操作码也完全相同。所以在ASL和LSL指令之间没有功能上的区别。6809在设计上之所以这样重复设置指令，是因为有些用户认为这种操作是算术左移，而另一些用户认为是逻辑左移。

在逻辑指令中还设有两种特殊的指令就是ANDCC和ORCC。这两条指令在6800系列中没有。由于设有这种指令，可以取代一些6800指令。在6800中对条件码寄存器进行操作的有8种指令。其中TAP和TPA是在累加器A和CCR之间传送数据的指令。在6809中可以用交换指令EXG和传送指令TFR来代替它们。6800中其余6种条件码寄存器进行置位和清零C、I和

V标志位的指令是：SEC、CLC、SEI、CLI、SEV、CLV。在6809中可以用ANDCC和ORCC这两条指令来代替它们。这两条指令使用立即寻址方式，数据字节可立即同条件码寄存器的内容进行逻辑与或者逻辑或操作实现对条件码寄存器CCR中任何一个标志位进行置位或清零。为了置位CCR中任一标志位，可以用逻辑1立即同该标志位进行逻辑或操作。为了清零CCR中任一标志位，可以用逻辑0立即同该标志位进行逻辑与操作。但如果进行或逻辑0操作或者与逻辑1操作时，标志位的状态则不会改变。所以当其它标志位需要改变时，用这种操作数可以保持不需改动的标志位的状态。

（2）逻辑指令操作码表

逻辑指令操作码表如表3.15所示。

（3）逻辑指令操作举例

清除条件码寄存器标志位之例

假设要清除条件码寄存器中的C和I标志位，确定所需指令和所用立即数字节。

由上可知，为了清除标志位,则该标志位需同逻辑0相与。所以,因为C和I标志位在第0和4位，故数据字节在这些位上应为逻辑0，而所有其它各位需为逻辑1，这样其它标志位

表3.14 6809逻辑指令

记忆符		操作	操作符号
AND	ANDA	A与存储器	A∧M→A
	ANDB	B与存储器	B∧M→B
	ANDCC	立即数与条件码寄存器	CC IMM CC
ASL	ASLA	A算术左移	A, B, M: C □ ← b_7 □□□□□□□□ b_0 ← 0
	ASLB	B算术左移	
	ASL	存储器算术左移	
ASR	ASRA	A算术右移	A, B, M: (b_7 保持) b_7 □□□□□□□□ b_0 → □ C
	ASRB	B算术右移	
	ASR	存储器算术右移	
COM	COMA	A变反（1→0；0→1）	$\overline{A}$→A
	COMB	B变反	$\overline{B}$→B
	COM	存储器变反	$\overline{M}$→M
EOR	EORA	异或A	A⊻M→A
	EORB	异或B	B⊻M→B
LSL	LSLA	A逻辑左移	A, B, M: C □ ← b_7 □□□□□□□□ b_0 ← 0
	LSLB	B逻辑左移	
	LSL	存储器逻辑左移.	
LSR	LSRA	A逻辑右移	A, B, M: 0 → b_7 □□□□□□□□ b_0 → □ C
	LSRB	B逻辑右移	
	LSR	存储器逻辑右移	
OR	ORA	A或存储器	A∨M→A
	ORB	B或存储器	B∨M→B
	ORCC	立即数或条件码寄存器	CC∨IMM→CC
ROL	ROLA	A循环左移	A, B, M: C □ ← b_7 □□□□□□□□ b_0 (循环)
	ROLB	B循环左移	
	ROL	存储器循环左移	
ROR	RORA	A循环右移	A, B, M: C □ → b_7 □□□□□□□□ b_0 (循环)
	RORB	B循环右移	
	ROR	存储器循环右移	

表 3.15　逻辑指令操作码

指令形式		6809寻址方式 固有 OP	~	#	直接 OP	~	#	扩充 OP	~	#	立即 OP	~	#	变址 OP	~	#	说明	5 H	3 N	2 Z	1 V	0 C
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+	A∧M→A	•	↕	↕	0	•
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+	B∧M→B	•	↕	↕	0	•
	ANDCC										1C	3	2				CC∧IMM→CC					1
ASL	ASLA	48	2	1														8	↕	↕	↕	↕
	ASLB	58	2	1														8	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+		8	↕	↕	↕	↕
ASR	ASRA	47	2	1														8	↕	↕	•	↕
	ASRB	57	2	1														8	↕	↕	•	↕
	ASR				07	6	2	77	7	3				67	6+	2+		8	↕	↕	•	↕
COM	COMA	43	2	1													A→A	•	↕	↕	0	1
	COMB	53	2	1													B→B	•	↕	↕	0	1
	COM				03	6	2	73	7	3				63	6+	2+	M→M	•	↕	↕	0	1
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+	A⊻M→A	•	↕	↕	0	•
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+	B∨M→B	•	↕	↕	0	•
LSL	LSLA	48	2	1														•	↕	↕	↕	↕
	LSLB	58	2	1														•	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+		•	↕	↕	↕	↕
LSR	LSRA	44	2	1														•	0	↕	•	↕
	LSRB	54	2	1														•	0	↕	•	↕
	LSR				04	6	2	74	7	3				64	6+	2+		•	0	↕	•	↕
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+	A∨M→A	•	↕	↕	0	•
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+	B∨M→B	•	↕	↕	0	•
	ORCC										1A	3	2				CC∨IMM→CC					7
ROL	ROLA	49	2	1														•	↕	↕	↕	↕
	ROLB	59	2	1														•	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+		•	↕	↕	↕	↕
ROR	RORA	46	2	1														•	↕	↕	•	↕
	RORB	56	2	1														•	↕	↕	•	↕
	ROR				06	6	2	76	7	3				66	6+	2+		•	↕	↕	•	↕

才不受与操作的影响。因此正确的数据字节应为11101110_2或EE_{16}。其指令应为：

```
ANDCC #
 EE
```

置位条件码寄存器标志位之例

假设要置位条件码寄存器中的N和F标志位，确定所需指令和所用立即数字节。

由上可知，为了置位标志位，则该标志位需同逻辑 1 相或。因为N和 F 标志位分别在第 3 和 6 位，所以数据字节在这些位上应为逻辑 1，而所有其它各位需为逻辑 0，这样其它标志位才不受或操作的影响。因此正确的数据字节应为01001000_2或48_{16}。其指令应为：

```
ORCC #
  48
```

指令解释之例

解释以下指令：

```
ANDCC #
  00
ORCC #
  10
ANDCC #
  FF
```

第一条指令是清除条件码寄存器，因为逻辑 0 同CCR所有各位相与。

第二条指令是置位I标志位，因为逻辑 1 同CCR中的I位相或，其它各位不变。

第三条指令除去消耗MPU时间之外，没有任何改变，因为逻辑1同CCR所有各位相与。

3. 测试指令

（1）测试指令的功能和操作

6809测试指令可以对以下数据进行测试：

- 存储器单元内容同A、B、D、X、Y、U和S寄存器进行算术比较。
- 存储器单元内容同累加器A和B进行逻辑比较。
- 对存储器单元、A和B进行零、正值或负值的测试。

6809中的测试指令和6800系列没有多大差别。6809的测试指令如表3.16所示。测试指令主要有三类：逻辑位测试（BIT）；算术比较测试（CMP）；零、正或负的字节测试（TST）。

位测试可以测试累加器A或B的单个数字位的状态（逻辑1或0），这时所有的数据屏蔽字节可以是指令语句的一部分内容（立即数）。或者是经由直接、扩充、变址寻址（可用任何一种变址寻址方式）的存储器单元内容。从表3.16可知，位测试是逻辑与操作，它们只影响CCR中的N和Z标志位。当用所有测试指令时，都没有“结果”形成，除去相应的置位或清除标志位之外。因为任何一种测试指令的主要功能是为了进行条件分支转移，进行简单的数据测试，只影响条件码寄存器中的某些标志位。

表3.16 测试指令

记忆符		操作	操作符号
BIT	BITA	位测试A	A∧M
	BITB	位测试B	B∧M
CMP	CMPA	同A比较	A−M
	CMPB	同B比较	B−M
	CMPD	同D比较	D−M:M+1
	CMPS	同S比较	S−M:M+1
	CMPU	同U比较	U−M:M+1
	CMPX	同X比较	X−M:M+1
	CMPY	同Y比较	Y−M:M+1
TST	TSTA	测试A	A−O
	TSTB	测试B	B−O
	TST	测试存储器单元	M−O

比较指令是使累加器A、B、D以及任何一个变址寄存器（X、Y、S、U）的内容同指令部分（立即）中的数据或者同使用直接、扩充以及任何一种变址寻址方式的存储器单元的内容相比较。比较操作是减法操作，用这种方法来决定两个数值是否相等、或者某个数值大于（小于）另一个数值。这些比较操作通常都用在条件分支转移指令之前，如相等转移（BEQ）、不等转移（BNE）、大于转移（BHI）、小于转移（BLO）等等。当要比较的数据同某个累加器或变址寄存器进行比较时，需从有关的累加器或变址寄存器中减去该数据，同时根据相应的指令使N、Z、V和C标志位置1或位0。而累加器或寄存器的内容，以及存储器单元的内容，都不会受这种操作的影响。6800中可以用一条固有指令（1字节）比较累加器B同累加器A。6809没有这种指令，然而任何一个16位寄存器内容（PC程序计数器例外），都可以和一个16位数据进行比较，以此作了弥补。

字节测试指令TST可以确定累加器A、B或任何一个存储器单元的内容是正、负还是零，而操作结果不影响有关寄存器的内容。这些指令通常用在条件分支转移指令之前，如正时转

移（BPL），负时转移（BMI）、等于零时转移（BEQ）、和不等于零时转移（BNE）。从表3.16可知，其操作是从有关的寄存器减去零。只有N和Z标志位受影响，字节测试操作时不产生新的结果。

（2）测试指令操作码表

测试指令操作码表如表3.17所示。

（3）测试指令操作举例

以下各例说明测试指令对条件码寄存器的作用，在下一节中还可看到它们同分支转移指令在一起的应用情况。

表3.17 测试指令操作码表

指令形式		6809寻址方式															说明	5	3	2	1	0
		固有			直接			扩充			立即			变址								
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	N	Z	V	C
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+	位测试A	•	↕	↕	0	•
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+	位测试B	•	↕	↕	0	•
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+	M同A比	8	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+	M同B比	8	↕	↕	↕	↕
	CMPD				10 93	7	3	10 B3	8	4	10 83	5	4	10 A3	7+	3+	M：M+1同D比	•	↕	↕	↕	↕
	CMPS				11 9C	7	3	11 BC	8	4	11 8C	5	4	11 AC	7+	3+	M：M+1同S比	•	↕	↕	↕	↕
	CMPU				11 93	7	3	11 B3	8	4	11 83	5	4	11 A3	7+	3+	M：M+1同U比	•	↕	↕	↕	↕
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+	M：M+1同X比	•	↕	↕	↕	↕
	CMPY				10 9C	7	3	10 BC	8	4	10 8C	5	4	10 AC	7+	3+	M：M+1同Y比	•	↕	↕	↕	↕
TST	TSTA	4D	2	1													测试A	•	↕	↕	0	•
	TSTB	5D	2	1													测试B	•	↕	↕	0	•
	TST				0D	6•	2	7D	7	3				6D	6+	2+	测试M	•	↕	↕	0	•

位测试指令应用例

假设要检查累加器A第6位逻辑状态，需要使用哪种测试指令？并求出正确的数据屏蔽字节。

为了测试累加器A的状态应该使用BITA指令，为检查第6位的状态，需使用的屏蔽字节为01000000_2或40_{16}。这里可知道屏蔽字节中的所有其它各位都用0来屏蔽掉。所以该指令是：

```
BITA #
  40
```

当执行上述指令时，如果累加器A的第6位为逻辑1，则实行与操作之后会使条件码寄存器的Z标志位清零，因为与操作结果不是0。如果上述操作结果Z标志位被置1，那么则说明累加器A第6位为逻辑0，因为与操作结果是0。

比较指令CMP应用例

试写出X寄存器的内容同某地址单元中的数据进行比较的程序，而该数据地址是在程序中比较指令之后50号（十进制）两个相邻的存储单元开始。

为实现上述任务所用指令是CMPX，因为操作数的地址指定的不是操作数本身，所以需使用间接寻址。另外因为操作数地址与相对于CMPX指令的位置有关，所以必须用程序计数器相对寻址。因此CMPX指令要采用程序计数器间接相对寻址。该指令为3字节：即CMPX操作码，后跟后缀字节，再后跟程序计数器相对偏值。因为CMPX指令需要3字节，包含操作数地址的存储单元应在离程序计数器的47_{10}号单元位置之处（不是50_{10}）。这是因为程序计

数器总指的是下一条要执行的指令。所以PC正确的偏值应是47_{10}或2 F_{16}。因此该操作的汇编语言操作码是：CMPX〔2 F，PCR〕。

从表3.4和表3.17可知，其指令码是：

CMPX指令操作码：AC

后缀字节：　　　9 C

相对偏值：　　　2 F

比较指令应用例

假设在以下指令执行时，累加器D的内容为0F50：

```
CMPD#
 OF
 49
```

确定在该指令执行之后，CCR标志位的状态。

执行CMPD指令时，将使0F50减去0F49，因为累加器D的内容大于比较操作中所用的数据。因此，N标志位被清零，表示结果为正。同时Z标志位也被清零，表示结果不为零。而且C标志位也将被清零，因为无进位或借位产生。在该操作时，其余标志位不受影响，并保持在前面操作后置1或置0的结果。

测试（TST）指令应用例

假设当遇到TSTA指令时，累加器A的内容为FF，确定在执行TSTA指令之后，CCR的状态。因为累加器A的F F内容是非零值，而且是负数（2的补码），所以Z标志位被清零，N标志位被置1。在该操作期间（见表3.17）V标志位总是为零。其它标志位不受该操作影响，将保持前面操作的结果，继续为1或0。

（4）其它算术、逻辑和测试指令操作

最后利用本节介绍的6809指令实现算术、逻辑和测试操作举例如下：

算术操作：

A加到B（A＋B→B）	PSHS　A ADDB，S++
X加到D（X＋D→D）	PSHS　X ADDD，S++
D加到X（D＋X→X）	LEAX D，X
Y加到X（Y＋X→X）	EXG　D，Y LEAX D，X EXG　D，Y
D减1（D－1→D）	EXG　D，X LEAX －1，X EXG　D，X
D加1（D＋1→D）	EXG　D，X LEAX 1，X EXG　D，X
D变负（$\overline{D}$＋1→D）	COMA COMB

	ADDD # 01
X变负 ($\overline{X}+1\rightarrow X$)	EXG D,X COMA COMB ADDD# 01 EXG D,X
D减X (D－X→D)	PSHS X SUBD ,S++
X减D (X－D→X)	PSHS D COMA COMB ADDD # 01 LEAX D,X PULS D

逻辑操作：

B与A (B∧A→A)	PSHS B ANDA ,S+
A与B (A∧B→B)	PSHS A ANDB ,S+
D算术左移	ASLB ROLA
D逻辑右移	LSRA RORB

测试操作

B位测试A (B∧A)	PSHS B BITA ,S+
A比较B (A－B)	PSHS B CMPA ,S+
B比较A (B－A)	PSHS A CMPB ,S+
X比较Y (X－Y)	PSHS Y CMP X ,S++

3.3.3 分支转移和其它指令

6809的最后两类指令是分支转移指令和其它指令。

代表6809判断能力强弱或是智能作用高低的指令是分支转移指令所具有的能力。这些指令通常用在算术、逻辑、或测试指令之后，根据运算操作结果，使条件码寄存器的各个标志位被

置 1 或置 0 的状态来进行操作。分支转移指令进行操作结果的判断，一定要根据条件码寄存器CCR的标志位的状态。当分支转移在执行开始时，程序计数器中被装入一个新的地址，所以使程序改变其执行方向，因此程序会继续根据分支转移操作所规定的新的地址开始执行。6809指令系统中有18条分支转移指令。每条指令都有短相对寻址和长相对寻址两种类型。

有关6809的其它指令的归类，是因为习惯上这些指令没有标准的分类法。这些指令是跳越转移JMP、跳越转移到子程序JSR、软件中断SWI、以及中断返回RTI等指令都属于这一类。这些指令中有许多是和6800系列相同的，但也设有一些很好的指令，6809有三种单独的软件中断指令，这样对最终用户是很有用的。6809的CWAI指令代替了6800的WAI指令。CWAI指令类似于WAI指令，但它是 2 字节的指令，可以清零条件码的任何一位标志位。最后，6809指令系统中还设有一条指令，它可以对外部硬件的过程与系统软件同步。该指令称为SYNC，这在那些大型系统中，特别是要求迅速传送数据的应用中，这种指令是很有意义的。

1. 分支转移指令

6809的分支转移指令如表3.18所示。所有的6809分支转移指令都是相对寻址方式。分支转移指令有两类：即无条件分支转移和条件分支转移。

（1）分支转移指令的功能和操作

无条件分支转移就是不管任何条件，程序都要进行分支转移（或者不进行分支转移）。6809中这些分支转移指令是分支转移BRA、向子程序分支转移BSR、不分支转移BRN指令。BRA和BSR指令，无论在什么条件下都使6809的程序分支转移到目的地址上。而BSR指令是调用子程序的指令，这时程序计数器的内容被保留在硬件 S 堆栈之中，以便实现向主程序的正常的返回。该指令类似于JSR指令。BRN指令是6800系列中没有的指令。该指令实际上是完成 2 个或 4 个字节的空操作（NOP）指令。此时6809经过该指令字节和相对地址偏值使用一定的周期时间，而不改变其执行内容。可以认为这条指令表面上没有意义，但为什么6809中又设置了这样一条指令呢？主要的思想就是：程序设计过程中，可以掩盖或隐藏指令操作码使其成为相对地址偏值。例如，6809首先经过BRN指令时，它就会把被掩盖的（或隐藏的）操作码当作相对地址偏值读出来，但不会产生操作结果。而在以后的程序中，可以跳回到被掩盖（隐藏）的操作码使其执行。这样做就可以节约程序设计的步骤，而且对记忆来说也是很有用的技窍。

6809其余的分支转移指令都是条件转移指令。条件分支转移指令需要取决于由条件码寄存器给出的某些条件。如果条件不满足，程序就继续执行下一条指令，不进行分支转移。条件分支转移指令通常是在测试、算术或逻辑运算指令之后被使用。然而这些指令也可以在任何一条对寄存器进行操作的指令之后使用，因为根据操作的结果，可以对条件码寄存器的某些标志位进行置 1 或置 0 。因为这是根据条件码寄存器标志位的状态，所以在遇到条件分支转移指令时，可能进行分支转移，也可能不进行分支转移。象BEQ、BNE、BCS、BCC、BVS和BVC这些条件分支转移指令，以及对条件码寄存器中的某些状态标志的检查都是很简单明显的。例如，BEQ或BNE是直接检查Z标志位的状态。因此可以把这些分支转移指令称为简单分支转移指令。其余的条件分支转移指令可以再细分为两类：一类为带符号的条件分支转移，另一类为无符号的条件分支转移。当两个补码数据操作时，使用带符号的分支转移指令；而当两个不是补码的数据操作时，使用无符号的分支转移指令。一般情况下，在LD、ST、INC、DEC、TST、CLR或COM指令之后，不使用无符号分支转移指令。

表 3.18 6809 分支转移指令

记忆符		操作	转移测试
BCS	BCS LBCS	进位位为1，转移	$C=1$
BEQ	BEQ LBEQ	寄存器内容等于存储器内容，转移	$Z=1$
BGE	BGE LBGE	带符号的寄存器内容大于或等于带符号的存储器内容，转移	$N\veebar V=0$
BGT	BGT LBGT	带符号的寄存器内容大于带符号的存储器内容，转移	$Z\vee(N\veebar V)=0$
BHI	BHI LBHI	无符号的寄存器内容大于无符号的存储器内容，转移	$C\vee Z=0$
BHS	BHS LBHS	无符号的寄存器内容大于或等于无符号的存储器内容，转移	$C=0$
BLE	BLE LBLE	带符号的寄存器内容小于或等于带符号的存储器内容，转移	$Z\vee(N\veebar V)=1$
BLO	BLO LBLO	无符号的寄存器内容小于无符号的存储器的内容，转移	$C=1$
BLS	BLS LBLS	无符号的寄存器内容小于或等于无符号的存储器内容，转移	$C\vee Z=1$
BLT	BLT LBLT	带符号的寄存器内容小于带符号的存储器内容，转移	$N\veebar V=1$
BMI	BMI LBMI	内容为负，转移（N标志位置1）	$N=1$
BNE	BNE LBNE	寄存器内容不等于存储器内容，转移（Z标志位置0）	$Z=0$
BPL	BPL LBPL	内容为正，转移	$N=0$
BRA	BRA LBRA	无条件转移	无
BRN	BRN LBRN	不转移	无
BSR	BSR LBSR	子程序转移	无
BVC	BVC LBVC	溢出标志位为0，转移	$V=0$
BVS	BVS LBVS	溢出标志位为1，转移	$V=1$

从表3.18中可以看到，带符号的或者无符号的分支转移指令使程序改变执行方向，是在条件码寄存器某些标志位进行逻辑组合的结果而进行的。

表3.19给出了刚才讨论的简单的、带符号的、无符号的三类条件分支转移指令。在该表中，对每一种分支转移指令都对应有其互补的（相反的）分支转移的条件。例如，BEQ的相反测试为BNE；BLT的相反测试为BGE等等。另外在该表中，把BEQ/BNE列入了所有三类条件分支转移。

（2）分支转移指令操作码表

分支转移指令的操作码表如表3.20所示。

表 3.19　6809 条件分支转移指令

简单条件分支转移指令	
条　件	相反条件
BEQ	BNE
BMI	BPL
BCS	BCC
BVS	BVC
带符号条件分支转移指令	
条　件	相反条件
BGT	BLE
BGE	BLT
BEQ	BNE
无符号条件分支转移指令	
条　件	相反条件
BHI	BLS
BHS	BLO
BEQ	BNE

（3）分支转移指令操作举例

BNE指令的应用

设有以下程序，它会产生何种动作？

```
 LDY  #
 FC
 50
┌→LEAY,-Y
│ CMPY
│ 00
│ 00
│ BNE
└─F 8
  ⋮
```

在该程序中，首先对Y装入立即数FC50，然后用LEAY指令使Y寄存器减1，在减1之后，Y寄存器中内容立即同0000相比较，这时进行比较操作就是从规定的寄存器（此时为Y）中减去操作数，根据结果使条件码寄存器某些标志位置1或置0。在本例中与Z标志位有关，只有当Y寄存器内容为0000时，Z位被置1。BNE指令在Z标志位被置1之前，总会使程序转移回到LEAY指令之处。所以程序在Y寄存器中的内容被减到零之前，总在循环。通过这种程序可以在程序之中建立时间延迟。在本程序中，BNE指令的相对地址偏值为F8，读者可

表 3.20 分支转移指令操作码表

指令形式		相对 OP	~[3]	#	说明	5 H	3 N	2 Z	1 V	0 C
BCC	BCC	24	3	2	C＝0，转	•	•	•	•	•
	LBCC	10 24	5(6)	4	C＝0，长转	•	•	•	•	•
BCS	BCS	25	3	2	C＝1，转	•	•	•	•	•
	LBCS	10 25	5(6)	4	C＝1，长转	•	•	•	•	•
BEQ	BEQ	27	3	2	Z＝1，转	•	•	•	•	•
	LBEQ	10 27	5(6)	4	Z＝1，长转	•	•	•	•	•
BGE	BGE	2C	3	2	≥0，转	•	•	•	•	•
	LBGE	10 2C	5(6)	4	≥0，长转	•	•	•	•	•
BGT	BGT	2E	3	2	>0，转	•	•	•	•	•
	LBGT	10 2E	5(6)	4	>0，长转	•	•	•	•	•
BHI	BHI	22	3	2	大于，转	•	•	•	•	•
	LBHI	10 22	5(6)	4	大于，转	•	•	•	•	•
BHS	BHS	24	3	2	大于等于，转	•	•	•	•	•
	LBHS	10 24	5(6)	4	大于等于，长转	•	•	•	•	•
BLO	BLO	25	3	2	小于，转	•	•	•	•	•
	LBLO	10 25	5(6)	4	小于，长转	•	•	•	•	•
BLS	BLS	23	3	2	小于等于，转	•	•	•	•	•
	LBLS	10 23	5(6)	4	小于等于，长转	•	•	•	•	•
BLT	BLT	2D	3	2	<0，转	•	•	•	•	•
	LBLT	10 2D	5(6)	4	<0，长转	•	•	•	•	•
BMI	BMI	2B	3	2	负，转	•	•	•	•	•
	LBMI	10 2B	5(6)	4	负，长转	•	•	•	•	•
BNE	BNE	26	3	2	Z＝0，转	•	•	•	•	•
	LBNE	10 26	5(6)	4	Z＝0，长转	•	•	•	•	•
BPL	BPL	2A	3	2	正，转	•	•	•	•	•
	LBPL	10 2A	5(6)	4	正，长转	•	•	•	•	•
BRA	BRA	20	3	2	转	•	•	•	•	•
	LBRA	16	5	3	长转	•	•	•	•	•
BRN	BRN	21	3	2	不转	•	•	•	•	•
	LBRN	10 21	5	4	不长转	•	•	•	•	•
BSR	BSR	8D	7	2	转子	•	•	•	•	•
	LBSR	17	9	3	长转子	•	•	•	•	•
BVC	BVC	28	3	2	V＝0，转	•	•	•	•	•
	LBVC	10 28	5(6)	4	V＝0，长转	•	•	•	•	•
BVS	BVS	29	3	2	V＝1，转	•	•	•	•	•
	LBVS	10 29	5(6)	4	V＝1，长转	•	•	•	•	•

自行验证。

BLT指令的应用

设有以下程序，它会产生何种动作？

```
      ⋮
  →CMPA,X+
 | BLT
 └FC
  LDB ,-X
      ⋮
```

该例是由X寄存器规定的存储器单元的内容同累加器A中的内容进行比较。只要带符号的累加器A的内容小于带符号的存储器内容时，BLT指令就将使程序返回到CMPA指令处。在例中CMPA指令采用自动加1方式使X寄存器内容加1，所以每执行一次循环，X寄存器内容都在增加。这样依次来检索存储器单元，直到找出小于或等于当前累加器A中内容的数值为止。一旦找到，它就被装入累加器B，以进行下面的处理。为装入准确的数值，LDB指令需采用X寄存器先减1的指令寻址，因为前面的CMPA指令对X寄存器做的是后加1寻址。

BLO指令的应用

在上例之中如果用BLO指令取代BLT指令，将会产生怎样动作呢？

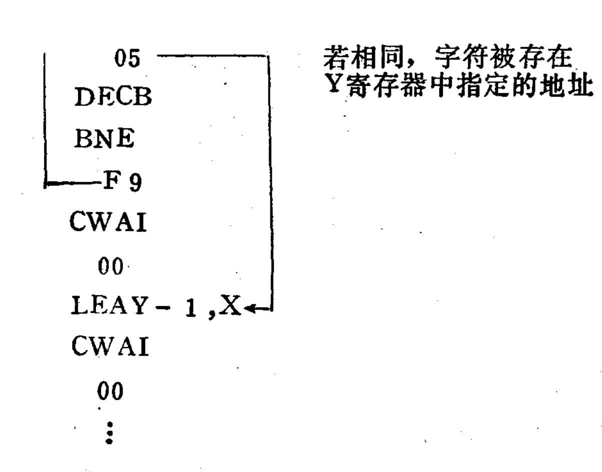
这时的区别仅在于6809不把累加器A和存储器内容作为两个带符号的补数考虑。而是只要无符号的累加器A内容小于无符号存储器内容即可出现分支转移。这样程序就会去查找存储器单元内容中小于或等于当前无符号的累加器A内容的数值。

检索字符应用

设用程序检索一个有32_{10}(20_{16})个字符的存储器表中所指定的字符，一旦找到该字符，即把它存在Y寄存器中规定的地址单元。又设该字符是“S”其ASC II编码为53_{16}。字符表的存储器起始单元为0100。

该字符检索程序如下：

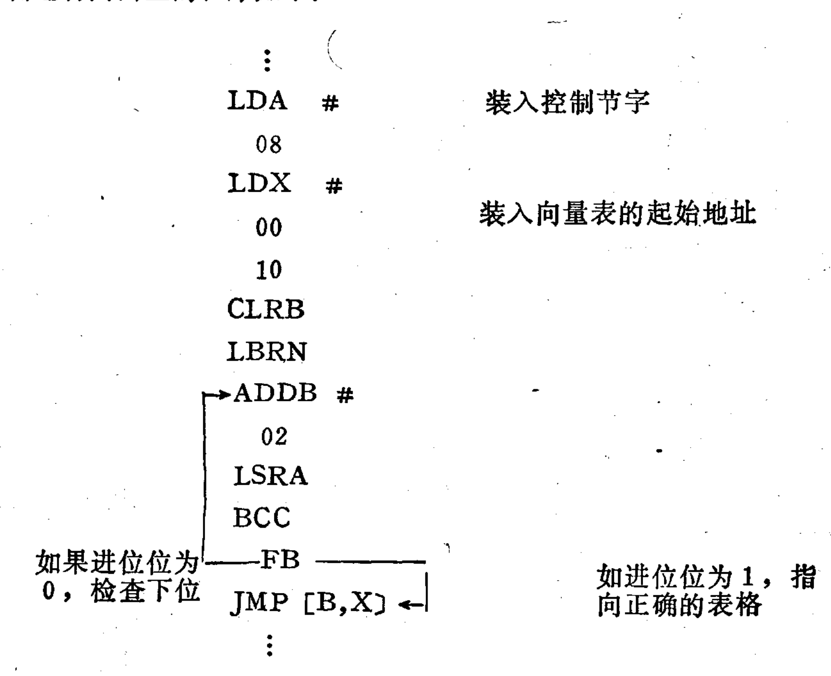
```
                         ⋮
                         LDX   #
                         53    #        装入所查找的字符
                         LDX
                         01             装入表格起始地址
                         00
                         LDB   #
                         20             装入表格长度
若不同，比较下个字符  →CMPA,X+          字符相同否？
                     |   BEQ
```



该程序无需规定指令的地址，所以其位置是独立的。X寄存器作为指示寄存器，采用自动加1的CMPA指令使X增加，以便在找到符合的数值之前，逐个给出表格数值。LEAY指令目的是把字符所存储的地址放在Y寄存器之中。在地址数值被传送到Y寄存器之前，该指令使X寄存器减1，这样做是因为程序在该点时，X指示寄存器内容在实际字符地址之前。

计算GO TO转移应用

假设希望指向一个向量控制字节表格，而控制字节只有一位为1。在控制字节中该位为1的位置确定表格中8个向量，从而使程序执行转移到该表格上面。具体设定是，控制字节的第3位为1，8个向量安排在起始地址为0100的16个存储单元（每个向量2个字节），所以要访问的向量被放在地址0106之中。按照以上要求，根据控制字节使程序的执行转移到相应的向量表，准确地访问向量的程序如下：



控制字节首先被装入累加器A，然后变址寄存器装入向量表的起始地址0100，为了确定正确的向量，使用累加器B作为常数偏值。每次使用LSRA和BCC指令对控制字节进行测试

之后，累加器B加2。因为控制字节的第3位是1，所以分支转移循环停止时，累加器B的内容为06_{16}。这样，在地址0100+06或0106中的向量将用JMP指令进行访问。因为JMP指令采用了间接寻址方式，所以程序控制将会转移到安排在地址0106中向量所指出的表格。在此应注意LBRN长不分支转移指令的意义，在第一次通过该程序时，ADDB指令不会被执行，因为此时该指令操作码变成了LBRN指令的相对地址偏值。但如果在LSRA指令之后C标志位为零时，该程序则会返回到被掩盖（隐藏）的操作码ADDB。这样做的结果，就使在开始装入X寄存器中向量表的起始地址，不是比起始向量表地址少2的地址。所以说这是使用BRN指令掩盖或者说隐藏指令操作码的一个极好的例子。

2. 其它类指令

6809指令系统中其余各指令，都归在其它类指令之中，因为它们不属于前面讨论过的任何类指令。这些指令如表3.21所示。表中许多指令都和6800系列中的相同，如跳越转移指令JMP、跳越转移到子程序指令JSR、空操作指令NOP、中断返回指令RTI以及子程序返回指令RTS。但，JMP和JSR指令现在可以用包括间接寻址在内的直接寻址、扩充寻址或任何一种变址寻址方式。在此只讨论6809中比6800增多的或改进的其它指令，它们是等待指令CWAI、软中断指令SWI 1、SWI 2、SWI 3和同中断同步的指令SYNC。

表 3.21 6809其它类指令

记忆符		操作	操作符号
CWAI		AND CC，后等待中断	CC∧IMM→CC 等待中断
JMP		跳越转移	EA→PC
JSR		跳越转移到子程序	无
NOP		空操作	无
RTI		中断返回	无
RTS		子程序返回	无
SWI	SWI 1	软中断 1	无
	SWI 2	软中断 2	无
	SWI 3	软中断 3	无
SYNC		同步中断	无

（1）CWAI指令

6809的等待中断指令CWAI和6800的等待中断指令相类似之点在于它使处理器处在等待中断的状态。但6809的CWAI是一个2字节指令，有指令操作码和数据字节。当CWAI指令被执行时，其数据字节要同条件码寄存器内容相逻辑与，与操作的结果放在条件码寄存器之中。所以这种操作就可以在进行堆栈操作之前来改变条件码寄存器中的内容。另外，如果CWAI的数据字节为00_{16}，这时条件码寄存器就被清零。当然如果确实知道在以后的中断服务程序中，不需要条件码寄存器的内容时，可以要求条件码寄存器清零。这样就可使服务程序在开始时就使用被清零的条件码寄存器。但如果CWAI的数据字节为FF_{16}时，条件码寄存器的内容将保持不变。

使用CWAI指令的事件顺序如图3.28所示。当6809遇有CWAI指令时，则条件码寄存器的内容与CWAI数据字节相与，结果放在条件码寄存器中。然后条件码寄存器的E标志位被置1，这是因为下面的操作要把所有的内部寄存器的内容放入S堆栈之中。而且前面也说明过，当E标志位为1时，通知6809在前面进栈操作中已经进栈的所有寄存器（S除外）同时也都要进行出栈。寄存器的进栈次序和前面讲过的PSH和PUL指令操作时相同。进栈顺序和所

有的自动进栈操作都完全相同。在内部寄存器被放在S堆栈之后，6809即进入等待循环。而等待循环，只要接受了四个硬件中断——$\overline{RESET}$、$\overline{NMI}$、$\overline{IRQ}$或$\overline{FIRQ}$中任何一个中断，即宣告结束。从图3.28中可知，为用$\overline{IRQ}$中断停止等待循环，条件码寄存器的I标志位须要清零，而为了用$\overline{FIRQ}$中断停止等待循环，条件码寄存器的F标志位须要清零。由于CWAI可以进行与逻辑操作，所以在停止等待循环过程中还可以进行某些控制动作。例如，如果恰好在CWAI指令之前使I标志位置1，并且使CWAI的数据字节为10用来保护I标志位的状态，这样就会防止IRQ中断来停止等待循环。同理，也可以用F标志位来防止$\overline{FIRQ}$来停止等待循环。另外还可以用CWAI指令来保证使$\overline{IRQ}$、$\overline{FIRQ}$或者两个同时都进行工作。例如，CWAI的数据字节为EF_{16}时，使$\overline{IRQ}$工作；为BF_{16}时，使$\overline{FIRQ}$工作；为AF_{16}时，使$\overline{IRQ}$和$\overline{FIRQ}$两者都工作。

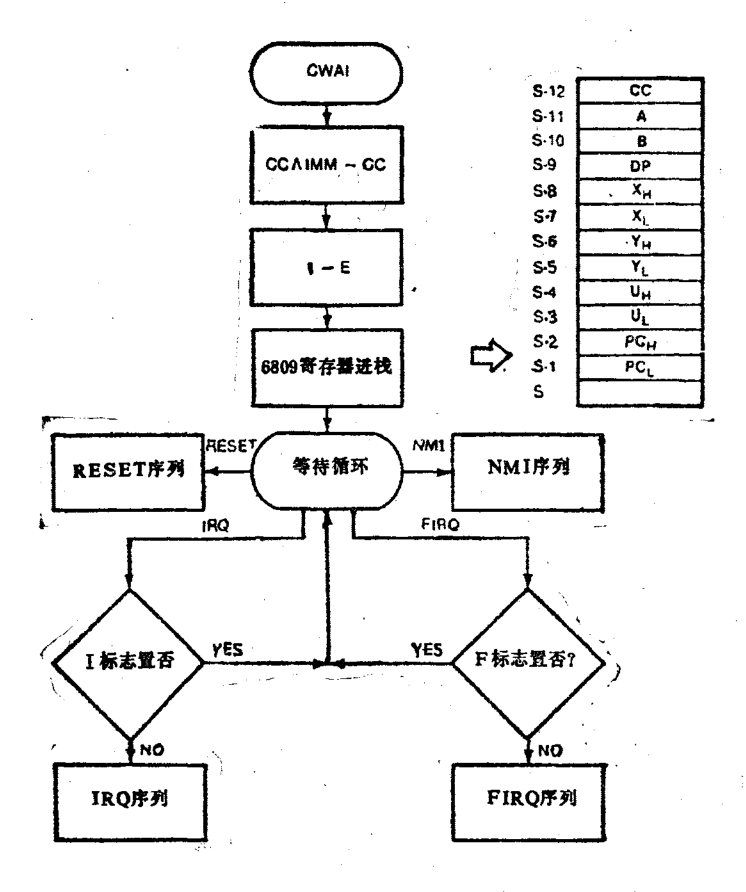


图3.28 CWAI事件的顺序

CWAI指令执行举例：

设有以下6809程序：

```
        :
      LDA  #
       40
      TFR A, CC
```

CWAI

40

要求确定在CWAI指令执行之后条件码寄存器的内容。

因为累加器A首先装入40_{16}，后把该内容传送到条件码寄存器，所以F标志位被置1，而其它CCR的标志位为0。CWAI的数据字节为40_{16}，在CWAI进行与操作期间将保护F标志位。这样$\overline{FIRQ}$中断就不会破坏等待循环。另外，CWAI操作结果，还要使E标志位为1。所以条件码寄存器的内容为11000000_2或$C0_{16}$。

如果在执行上述程序之前S寄存器的内容是$E600_{16}$，那么条件码寄存器内容可在S堆栈的什么地址上找到呢？从图3.28可知CCR的内容所在地址为$E600_{16}-12_{10}$，或$E5F4_{16}$。

（2）SWI1、SWI2、SWI3指令

6809中有三级软件中断指令，即SWI1、SWI2和SWI 3。执行它们时可以使处理器指向有关的中断服务程序。软中断在程序调试过程中通常在程序中作为插入的断点使用，而且软中断在单步程序、操作系统调用以及软件开发系统中都很有用。另外，硬件中断还能用软中断来仿真。在6809中有三级软中断，原因是在6800系列中只设有一级软中断，而且它往往被用在ROM的监控程序之中，所以对最终用户没有意义，实际上最终用户不能使用它。但又不象商用软件包那样要用到所有的6809三级软中断，因此在这种情况下至少要求有一个软中断留给用户系统使用。莫托罗拉公司已约定不使用SWI2软中断，而且建议所有开发6809软件包的系统公司这样做，为的是留给用户使用。

6809三个软中断的优先顺序是：SWI1、SWI2和SWI3。这三个软中断的受到事件作用后的处理顺序如图3.29所示。从图中可以看到，当前指令一旦完成，条件码的E标志位即被置1，表示6809所有寄

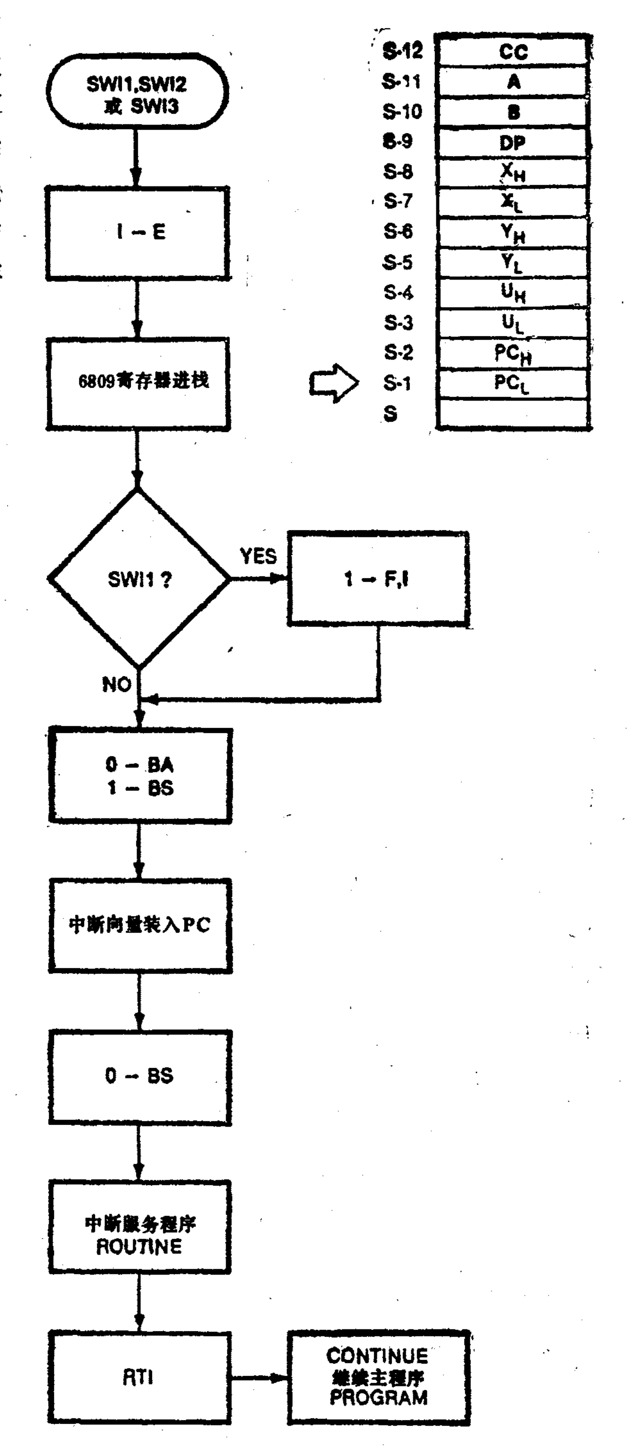


图3.29 SWI1、SWI2和SWI3的事件处理顺序

存器(S除外)将按图3.29中所示的顺序进入S堆栈区之中。在进栈之后，如果执行的是SWI1软中断，则需使F和I标志位置1，以防止在SWI1中断服务程序期间可能出现的任何$\overline{\text{FIRQ}}$或$\overline{\text{IRQ}}$硬件中断。如果执行的是：SWI2或SWI3软中断，就不去屏蔽$\overline{\text{FIRQ}}$和$\overline{\text{IRQ}}$硬件中断，除非在中断服务程序期间使F或I标志位置1。然后6809确认接受软中断，给外部装置的表示是在总线状态输出线BS上给出高电平（该线含意前面讲过）。接着在程序计数器中装入相应的中断向量，BS回到低电平后即开始执行对应的中断服务程序。一旦完成服务程序，程序即直接回到主程序，此时在中断服务程序的最后一条指令是RTI指令即可实现此要求。在RTI指令执行时，自动地使原来寄存器的内容从S堆栈中出栈，使6809返回到前面的状态。

如前所述，软件中断或任何其它中断都会使6809指向相应的中断服务子程序。中断向量就是各个中断服务子程序的起始地址。6809中每个软件或硬件中断都设有一个唯一的向量地址，它们分配在存储器的最后16个单元（FFF0～FFFF）。

6809的中断向量所在地址分配情况如表3.22所示。其中每个中断向量都设有一对地址。这些地址单元通常都在ROM之中，所以是固定的用户自已不能改变。因为6809要根据这些地址来取出向量后，转到另外的地址上执行，所以有时又把中断向量称为绝对间接寻址。在表3.22中也给出了每个中断的相对优先权，$\overline{\text{RESET}}$的优先权最高。同时还可以看出对向量地址FFF0:FFF1作了保留，可以为其它应用作好准备。

表 3.22　6809中断向量的存储地址分配

向量单元	指定向量	相对优先权
FFF0:FFF1	保　留	低
FFF2:FFF3	SWI3	
FFF4:FFF5	SWI2	
FFF6:FFF7	$\overline{\text{FIRQ}}$	
FFF8:FFF9	$\overline{\text{IRQ}}$	
FFFA:FFFB	SWI1	
FFFC:FFFD	$\overline{\text{NMI}}$	↓
FFFE:FFFF	$\overline{\text{RESET}}$	高

（3）SYNC指令

SYNC是中断同步指令，使用该指令可以用外部的硬件事件来同步系统软件，如象I/O装置的快速数据传送。当执行程序遇有SYNC指令时，所有执行都被暂停，而6809进入等待中断循环，这就是同步状态。在同步状态期间，6809的地址线和数据线都处在高阻抗(三态)状态，或者说处在与外部总线断开的状态。当前若发生硬件中断，则会出现两种情况：

· 如果中断没被屏蔽，而且作用在3个或3个以上的MPU周期时，则6809将中断等待循环并执行相应的中断服务程序；

· 如果中断被屏蔽，或作用的有效性少于3个MPU周期，则6809就会继续执行主程序，即执行下面的指令。

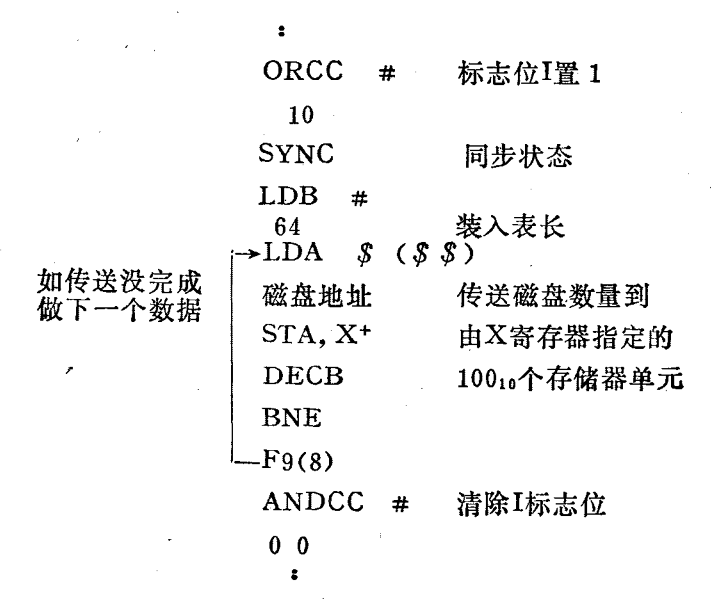
第一种情况下，SYNC指令的性质，除去不使内部寄存器的数据进栈之外，非常象CWAI指令的作用，在这种情况下，如果中断没有被屏蔽，任何正常的中断都会破坏同步状态。

但在第二种情况下，就可用SYNC指令来使外部硬件过程同步主程序。我们知道，使用中断的主要问题（缺点）是它们不能与程序的执行同步，换句话说，在主程序执行时无法安排中断或者在主程序之内的某个时间上发生中断。而SYNC指令可以满足使用中断的要求，

并在相同的时间内安排中断以达到系统的最佳效率。例如象高速磁盘这种输入/输出装置，可以使用6809中断请求线（$\overline{IRQ}$或$\overline{FIRQ}$），当使该线工作时，表示磁盘为传送数据做好了准备。在SYNC指令之前，可以先分别把中断屏蔽位置1，（I或F位）以屏蔽掉正常的中断请求。但输入/输出装置选择线在工作时，使程序做的是下面要执行的程序而不是对于该中断所要执行的正常的服务程序。所以下面的指令就会开始执行数据传送过程，并继续执行主程序。因此在硬件和软件两方面进行了同步，而在时间上也是节约的，因为这时没有寻找向量或进栈的问题。

SYNC指令应用举例：

要求在程序的某个特定点，从磁盘机往存储器装入100_{10}字节的数据，实现该过程的程序如下：



假设磁盘机使用$\overline{IRQ}$线表示数据传送。第一条指令是置条件码寄存器中的I标志位为1，以便屏蔽任何正常的$\overline{IRQ}$中断。然后SYNC指令在磁盘机使$\overline{IRQ}$线工作之前暂停主程序的执行。当该线工作时，主程序使用X寄存器按自动加1变址寻址方式把100_{10}(64_{16})个数据字节存储到100_{10}个相应的存储单元中去。然后主程序清除I标志位，并继续做下去。如若在同步状态期间，其它某个硬件中断（$\overline{NMI}$、$\overline{FIRQ}$、$\overline{RESET}$）造成了中断的条件，那么就有可能破坏同步状态。

（4）其它类指令操作码表。

其它类的指令操作码表如表3.23所示。

3.3.4 6800的等效指令

我们知道在6809的指令系统中，许多6800指令的记忆符都不存在，为了在6809系统中可以运行6800软件，在下面给出把6800指令翻译成功能上等效于6809操作的指令，如表3.24所示。

表3.23 其它类指令操作码

指令形式	固有			直接			扩充			立即			变址			相对			说明	5	3	2	1	0
	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	N	Z	V	C
CWAI	3C	20	2																CC∧IMM→CC 等待中断					1
JMP				0E	3	2	7E	4	3				6E	3+	2+				EA→PC	•	•	•	•	•
JSR				9D	7	2	BD	8	3				AD	7+	2+				跳转子	•	•	•	•	•
NOP	12	2	1																空操作	•	•	•	•	•
SWI SWI	3F	19	1																软中断1	•	•	•	•	•
SWI2	10 3F	20	2																软中断2	•	•	•	•	•
SWI3	11 3F	20	2																软中断3	•	•	•	•	•
SYNC	13	>2	1																同步中断	•	•	•	•	•
RTI	3B	6/15	1																中断返回					7
RTS	39	5	1																子程序返回	•	•	•	•	•

表3.24 6800等效指令

6800 指令	6809 等效
ABA	PSHS B; ADDA ,S+
CBA	PSHS B; CMPA ,S+
CLC	ANDCC #FE
CLI	ANDCC #EF
CLV	ANDCC #FD
CPX	CMPX
DES	LEAS -1,S
DEX	LEAX -1,X
INS	LEAS 1,S
INX	LEAX 1,X
LDAA	LDA
LDAB	LDB
ORAA	ORA
ORAB	ORB
PSHA	PSHS A
PSHB	PSHS B
PULA	PULS A
PULB	PULS B
SBA	PSHS B; SUBA ,S+
SEC	ORCC #01
SEI	ORCC #10
SEV	ORCC #02
STAA	STA
STAB	STB
TAB	TFR A,B; TST A
TAP	TFR A,CC
TBA	TFR B,A; TST A
TPA	TFR CC,A
TSX	TFR S,X
TXS	TFR X,S
WAI	CWAI #FF

3.4 6809的软件设计技术

3.4.1 概述

在最近一段时期中，新的程序设计技巧不断在发表，因此，象ALGOL语言中的IF-THEN-ELSE语句和WHILE-DO语句等所构成的软件性能要重新进行评价。

6809微处理器的软件结构和指令系统适合于结构化程序设计和再入程序设计，这一点在第一章已作过说明。在6809系统中易于实现的新的程序设计技巧有以下五种：

- 位置独立型程序设计
- 再入型程序设计
- 递归型程序设计
- 协同型程序设计
- 全变量、局部变量处理方法（堆栈区作业）

为了清楚地了解这些程序设计技巧或方法，在本节中将举出具体实例进行说明以餍读者。同时在最后内容中，对软中断的应用也给予一定的说明。

3.4.2 位置独立型程序设计

位置独立型程序设计的基本技巧就是：不要访问绝对地址，但需对所设置的数据或分支转移的目的地址，在程序执行的流程中进行控制。

例如，某个要执行的程序写在地址＄2000到＄2FFF之中。如果需要依次序执行，但因硬件系统资源有限，只有地址＄A000区为空白区，这时如要求程序使用绝对地址，把＄A000区作为程序区，那么就必须对程序再次进行汇编。但是，如果程序已经被固化在ROM中时，那么也就不能执行了。另外，如果程序本身是用机器语言编写的，那么定位这些程序还要花更长的时间。但如果使用位置独立型程序设计，则可原封不动地从＄A000地址开始执行。

因此对于象"该程序从何地址到何地址、为几K字节、起始在何地址"等这些规定的项目，也就没有必要翻来复去地研究。而是把用位置独立技巧编写的模块化程序放在ROM或外存储器之中，利用相应的程序把ROM或外存储器中的模块化程序传送到应该执行的位置（无论传送到存储器哪个空间）之上，这时就可以立即执行。从这个意义上讲，位置独立型程序又可称为定位型程序。因此不管什么程序，使用6809都可写成位置独立型程序，以下将具体举例说明。

1. 绝对寻址方式的应用限制

在使用位置独立型程序设计技巧时，绝对寻址方式的应用要受到限制，只有在由硬件设计来决定地址的输入/输出口的处理上，或者在监控程序中固化了的子程序，如监控程序中的输入/输出子程序等的应用上，可以使用绝对寻址方式。

所以，除此之外，几乎全部子程序的调用都可使用BSR或者LBSR。而对于无条件跳越转移可以使用BRA或LBRA指令。因为这些指令都使用相对寻址——即相对于当前程序计数器的数值就可找到指令或数据的地址，因而符合位置独立型程序设计的基本原则。对于使用JSR、JMP等绝对地址跳转的指令，只要把程序计数器作为指示寄存器的变址寻址方式或间接变址寻址方式，使绝对寻址变为相对寻址，这种指令也可以使用。换句话说，除去调用监控程序之外的所有分支转移，只要使用分支转移子程序调用指令（BSR、LBSR）、或分支转移指令（BRA、LBRA）即可编出程序。

在表3.25对PIA（并行接口器件）的测试程序中，PIA的地址是＄8000，已为硬件固定

表3.25 测试PIA程序清单

(a) 位置固定程序

```
00007A 1000 BD   11F7      A START  JSR    TEST     ABSOLUTE ADDRESS
00008A 1003      01F4      A        BSZ    500      500 BYTES = $00

00010                        * SUBROUTINE TEST

00012A 11F7 7D   8000      A TEST   TST    PIA
00013A 11FA 26   FB   11F7          BNE    TEST
00014A 11FC 39                      RTS
```

(b) 位置独立程序

```
00007A 1000 17   01F4 11F7 START    LBSR   TEST     LONG RELATIVE
00008                      *                        ADDRESSING MODE
00009A 1003      01F4      A        BSZ    500      500 BYTES = $00

00011                        * SUBROUTINE TEST

00013A 11F7 7D   8000      A TEST   TST    PIA
00014A 11FA 26   FB   11F7          BNE    TEST     WAIT UNTIL
00015                      *                        PIA IS CHANGED TO  ZERO
00016A 11FC 39                      RTS
```

好的地址，也就是说所给的地址是绝对地址，所以，在使用PIA时，不能采用程序计数器相对寻址方式（程序计数器作为指示寄存器的变址寻址方式）。

如果对子程序的调用采用位置独立型程序设计，可使用相对寻址的分支转移子程序指令BSR，当超过－128～＋127字节时，可以使用LBSR这种分支转移子程序指令，不能使用JSR。

2. 立即寻址方式应用的限制

在位置独立型的程序设计中，采用立即寻址方式访问地址数值或跳越转移表的表格起始地址时，要考虑立即寻址方式使用的限制。例如表3.26所示程序之例。

表3.26 立即寻址方式程序

```
        LDX   #TABLE
        LDY   A, X
TABLE   FDB   START, STOP, BACK
START   EQU   *
STOP    EQU   *
BACK    EQU   *
```

其中指令LDX　#TABLE含意是把TABLE表示的起始地址装入X寄存器之中。但是，用TABLE来表始起始地址，并不会使程序由于所装入的不同形式的地址而有什么区别。也就是说，不是位置独立型的程序。因此如果把LDX　#TABLE指令改为LEAX　TABLE, PCR这种使用程序计数器相对寻址的指令，这样表格的起始地址即与程序装入的位置没有关系，照样可以进行访问。这时TABLE的数值不是偏值，而是指有效地址（和分支转移指令对操作数指定的方法相同）。

因此，设计位置独立型程序的问题，就是怎样使用程序计数器相对寻址方式问题的这种说法，并没有言过其实。

特别是LEA…　……，PCR这种对有效地址装入的指令，在执行访问表格时是非常重要的。表3.27所列程序，就是利用LEAX　TBALE, PCR的数据块传送程序。

表3.27 数据块传送程序

(a) 位置固定程序

```
00009            2000      A BLOCK  EQU    *
00010A 2000 8E   200C      A        LDX    #TABLE    IMMEDIATE ADDRESSING
00011A 2003 A6   80        A BLOCK1 LDA    ,X+
00012A 2005 A7   A0        A        STA    ,Y+
00013A 2007 81   04        A        CMPA   #4        END OF DATA ?
00014A 2009 26   F8   2003          BNE    BLOCK1
00015A 200B 39                      RTS
```

* LDX　#TABLE的操作数使用的是$200C绝对地址

(b) 位置独立程序

```
00009            2000      A BLOCK  EQU    *
00010A 2000 30   8C 09              LEAX   <TABLE,PCR PC RELATIVE ADDRESSING
00011A 2003 A6   80        A BLOCK1 LDA    ,X+
00012A 2005 A7   A0        A        STA    ,Y+
00013A 2007 81   04        A        CMPA   #4        END OF DATA ?
00014A 2009 26   F8   2003          BNE    BLOCK1
00015A 200B 39                      RTS
```

* LEAX<TABLE, PCR设使用绝对地址

3. 计算GO TO

位置独立型程序实现的根本要求是没有绝对地址的程序。在表3.28计算 GO TO的程序中，JTABLE表内的数据可以使用绝对地址，写为FDB LESS, FDB ZERO。但在COMPUT输入之后的JMP〔A, X〕中X的内容存放的是表格的起始地址＄2000，所以采用累加器A作为偏值，并按间接寻址方式即可直接转移到各个程序上。

即使把程序的位置移动到＄8000处，程序内数据、表格地址的内容也可不要改变。

所以，使用位置独立技巧设计程序时，象在JTABLE表中的情况应是：当从JTABLE起始地址来执行的程序需要转到"LESS"地址时，可在程序中预先放置好到"LESS"的偏值。COMPUT输入后，当把该偏值装入到累加器D时，即可转移到根据累加器D的内容和JTABLE起始地址进行累加器为偏值的变址寻址方式而得到的有效地址上面。因此，即使把程序位置从＄2000移到＄8000时，程序内容也完全没有变化。

计算GO TO这种程序的实际应用见第五章中8080仿真程序设计。

6809程序中有关表格中各个数据块的起始地址的分度点，可以用乘法指令计算出来，如以下所示的程序例，在累加器A中放入的内容是数据块的地址，所以访问表格时可使用这种计算方法。当然，使累加器内容加倍时，不用乘法，使用移位指令会更迅速些，程序也较短。

```
LDA   DATA, PCR     读取表格的起始地址
LDB   #SIZE         数据块的尺寸
MUL
LEAX D, X
```

4. 引入协同程序

协同程序的详细含意将在下面3.4.5节中介绍，本节只说明协同程序在位置独立型程序设计中的作用。

当使用分支转移子程序调用指令时，程序计数器的内容就会保留在系统堆栈之中。因此，可以把被保留的程序计数器的内容来作为数据的起始地址，从而实现位置独立型程序。

在本例之中，串行接口器件MC6850的绝对地址是＄FCF4/FCF5，其打印输出子程序如表3.29。

PRINTX子程序作为协同程序来执行时，其数据的起始地址是在调用PRINTX的主程序中被保留在系统堆栈内的程序计数器内容。

PRINTX协同程序使用变址寄存器X自动加1功能来取出打印数据。该数据采用扩充寻址的绝对寻址方式输出到串行口器件ACIA之中。

取出的数据为＄00时，表示数据结束，则从接在数据区的主程序"CONTNU"进入主程序。因为取出的数据为＄00时表示要"CONTNU"，所以要把变址寄存器X内容传送到程序计数器PC，因而进入主程序的"CONTNU"。

该程序是没有使用程序计数器相对寻址（把程序计数器作为指示寄存器用的变址寻址方式）的位置独立型程序。这时要把系统堆栈中的程序计数器内容传送给变址寄存器；在返回主程序时再把变址寄存器的内容传送给程序计数器。

表3.28 计算GO TO程序

(a) 位置固定程序

```
00007                          * JUMP TABLE

00009            2000     A JTABLE EQU    *
00010A 2000      200C     A        FDB    LESS      <
00011A 2002      2020     A        FDB    ZERO      =
00012A 2004      2034     A        FDB    GREATE    >

00014                      * COMPUTED GO TO ENTRY POINT
00015                      * ACC A CONTENTS THE STATUS
00016                      *               0 = LESS THAN
00017                      *               1 = EQUAL OR ZERO
00018                      *               2 = GREATER THAN

00020            2006     A COMPUT EQU    *
00021A 2006 48                     LSLA             A:=A*2

00023                      * FETCH JUMP TABLE (JTABLE) TOP ADDRESS
00024A 2007 8E   2000     A        LDX    #JTABLE

00026                      * COMPUTE EFFECTIVE ADDRESS
00027A 200A 6E   96       A        JMP    [A,X]

00029                      * DUMMY PROGRAM

00031            200C     A LESS   EQU    *         LESS PGM START HERE
00032A 200C      0014     A        BSZ    20
00033            2020     A ZERO   EQU    *         EQUAL / ZERO
00034A 2020      0014     A        BSZ    20
00035            2034     A GREATE EQU    *         GREATE PROG
00036A 2034      0014     A        BSZ    20
```

(b) 位置独立程序

```
00007                          * JUMP TABLE

00009            2000     A TABLE  EQU    *
00010A 2000      000E     A        FDB    LESS-TABLE <
00011A 2002      0022     A        FDB    ZERO-TABLE =
00012A 2004      0036     A        FDB    GREATE-TABLE >

00014                      * COMPUTED GO TO ENTRY POINT
00015                      * ACC A CONTENTS THE STATUS
00016                      *               0 = LESS THAN
00017                      *               1 = EQUAL OR ZERO
00018                      *               2 = GREATER THAN

00020            2006     A COMPUT EQU    *
00021A 2006 48                     LSLA             A:=A*2

00023                      * FETCH TABLE TOP ADDRESS
00024A 2007 30   8C F6             LEAX   TABLE,PCR

00026                      * FETCH OFFSET DATA FROM TABLE
00027A 200A EC   86       A        LDD    A,X       FETCH OFFSET

00029                      * COMPUTE EFFECTIVE ADDRESS
00030                      * THEN GO TO EFFECTIVE ADDRESS
00031A 200C 6E   8B       A        JMP    D,X

00033                      * DUMMY PROGRAM

00035                      * LESS THAN PGM START HERE
00036            200E     A LESS   EQU    *
00037A 200E      0014     A        BSZ    20

00039                      * EQUAL OR ZERO PGM START
00040            2022     A ZERO   EQU    *
00041A 2022      0014     A        BSZ    20

00043                      * GREATER THAN  PGM START
00044            2036     A GREATE EQU    *
00045A 2036      0014     A        BSZ    20
```

表3.29　打印输出子程序

```
00004                         * HARDWARE ADDRESS

00006           FCF4     A ACIAS  EQU    $FCF4     ACIA STATUS ADDRESS
00007           FCF5     A ACIAD  EQU    ACIAS+1   ACIA DATA ADDRESS

00009A 2000                       ORG    $2000

00011           2000     A PRINT  EQU    *
00012A 2000 8D  23   2025         BSR    PRINTX    PRINT ONE BLOCK CALL

00014                         * PRINT DATA AREA

00016           2002     A DATA   EQU    *
00017A 2002     P0       9        FCC    'PRINT ONE LINE'
00018A 2010     0D       A        FCB    $D,$A
00019A 2012     20       A        FCC    ' SAMPLE PROGRAM '
00020A 2022     0D       A        FCB    $D,$A
00021A 2024     00       A        FCB    0         END OF DATA FLAG

00023           2025     A CONTNU EQU    *


00025                         * PRINT ONE BLOCK SUBROUTINE

00027A 2025 35  10       A PRINTX PULS   X         RESTORE RETURN ADDRESS
00028A 2027 A6  80       A PRNT0  LDA    ,X+       FETCH PRINT DATA
00029A 2029 27  0C   2037         BEQ    PRNTZZ    END OF DATA
00030A 202B F6  FCF4     A PRNT1  LDB    ACIAS     READ ACIA STATUS
00031A 202E C5  02       A        BITB   #2        TEST TDRF
00032A 2030 27  F9   202B         BEQ    PRNT1     WAIT TDRE
00033A 2032 B7  FCF5     A        STA    ACIAD     SEND DATA TO LINK
00034A 2035 20  F0   2027         BRA    PRNT0
00035A 2037 1F  15       A PRNTZZ TFR    X,PC      RETURN TO CONTNU
```

上述各例之中的位置独立型程序都是按照相对寻址或程序计数器相对寻址方式，或者采用程序计数器和其它的指示寄存器之间进行交换的传送操作而完成的。

3.4.3　再入型程序设计

几个用户共享存储器中同一个程序，但仍可以执行的程序称为再入型程序。再入程序是存储器中所装入的程序模块（机器字程序模块）可具备的一种性质。也就是说，在同一个程序模块中，可以运行二个以上的多个任务，因此该程序具备并行使用性质，故称再入型程序设计。这在中断系统中是很重要的问题；特别是在大型程序中，这种方法可以节约大量的存储空间。堆栈操作对再入程序是最方便不过的，而6809设有两个堆栈，并且，需要时X、Y寄存器也可以作为堆栈指示器使用。

使用再入程序设计技巧编制的程序，不管是主程序还是中断处理程序都可以共同使用。对于不准许再入的普通程序，如果在主程序使用该程序当中发生了中断，则在中断处理时，按主程序进行处理的中间结果，将会由于中断的处理而完全被破坏，使之不能进行下去，如图3.30所示。

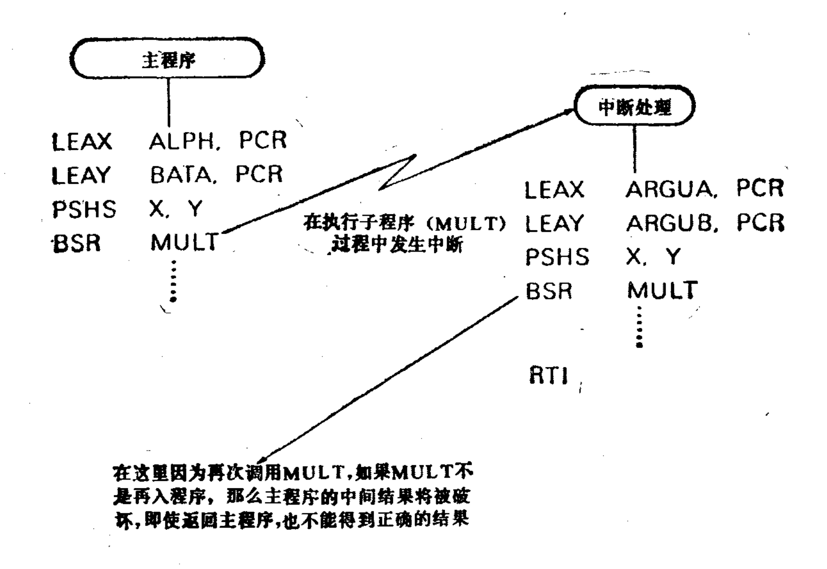


图3.30 再入程序实际过程

编制再入程序的原则是：MPU内的寄存器用系统堆栈指示器S统一管理起来，系统（硬件）堆栈区以外的区域不作为工作区使用。

即使有一个字节的数据被保留在堆栈区以外的存储区中，因该数据的存在，那时它就完全失去作为再入程序的功能。

现在以BCD（二-十进制数）数变换为BIN数（二进制数）的二个程序为例来比较说明。在第一个程序中：输入数据TW，结果数据RES各为 2 字节数据，使用堆栈区以外 3 个字节RAM作为运算使用。那么在表3.30的第一个程序中，由于使用了堆栈以外的RAM区，所以不能由两个以上的程序调用。但在第二个程序中，作为输入数据存在累加器D、输出数据也放在累加器D之中。完全没有使用堆栈以外的RAM区。在MULTBU子程序中的局部变量保证有 3 个字节在堆栈之内执行。在BCDBIN子程序中，输入数据、位数计数器缓冲区也都保证在堆栈之内。所以该程序不但是再入型程序，而且也是位置独立型程序。

因为6809和堆栈之内的运算处理能力很强，所以比起一般程序来说，再入程序方法的处理时间缩短。同时，在利用堆栈指示器的变址寻址方式中，几乎所有情况下后缀字节为一个字节，和操作码一个字节在一起共二个字节，所以使用一条指令即可执行。在基于绝对寻址方式的第一个程序中，除去用直接寻址方式写的程序外，由于都需要二字节的操作数，所以，每个存储器操作指令都增加了一个字节；而且还必须分配在堆栈区之外的RAM，因而这种程序是不太经济的。

在第二个程序中出现的LEAS 3，S 或LEAS-3，S堆栈本身的操作，将在3.4.6节中进行说明。

表3.30 BCD→二进制数变换子程序

(a) 非再入型程序

```
00013A 2000        0002   A TW      RMB    2         BCD DATA IN
00014A 2002        0002   A RES     RMB    2         BINARY DATA OUT
00015A 2004        0001   A CNT     RMB    1         DIGIT COUNT BUFFER
00016A 2005        0002   A TEMP    RMB    2         TEMPORARY DATA BUFFER

00018              2007   A BCDBIN  EQU    *
00019A 2007 86     04     A         LDA    #4        SET DIGIT COUNTER
00020A 2009 B7     2004   A         STA    CNT
00021A 200C CC     0000   A         LDD    #$0
00022A 200F FD     2005   A         STD    TEMP      CLEAR TEMPORARY BUFFER
00023A 2012 FD     2002   A         STD    RES       CLEAR RESULT

00025A 2015 8D     13   202A LOOP   BSR    SHIFT     FETCH 1 DIGIT
00026A 2017 FC     2005   A         LDD    TEMP      ACC D :=(TEMP)
00027A 201A F3     2002   A         ADDD   RES       ACC D :=ACC D + (RES)
00028A 201D FD     2002   A         STD    RES       RES := ACC D "SAVE D"
00029A 2020 7A     2004   A         DEC    CNT       CNT := CNT -1
00030A 2023 27     04   2029        BEQ    CMPLT     COMPLETED ?
00031A 2025 8D     1F   2046        BSR    MULT      CALL RES:=RES*10
00032A 2027 20     EC   2015        BRA    LOOP

00034A 2029 39              CMPLT   RTS

00036                       *
00037                       * SUBROUTINE SHIFT
00038                       *

00040A 202A C6     10     A SHIFT   LDB    #$10      SETUP FOR 4 BIT SHIFT
00041A 202C B6     2000   A         LDA    TW        READ TW
00042A 202F 3D                      MUL              4 BIT SHIFT LEFT
00043A 2030 B7     2006   A         STA    TEMP+1    SAVE TEMP RESULT (UPPER)
00044A 2033 F7     2000   A         STB    TW        SAVE LOWER BYTE
00045A 2036 C6     10     A         LDB    #$10      SETUP FOR 4 BIT SHIFT
00046A 2038 B6     2001   A         LDA    TW+1      READ LOWER TW BYTE
00047A 203B 3D                      MUL              EXEC 4 BIT SHIFT
00048A 203C F7     2001   A         STB    TW+1      STORE LOWER BYTE
00049A 203F BB     2000   A         ADDA   TW        ADD UPPER BYTE
00050A 2042 B7     2000   A         STA    TW        STORE UPPER BYTE
00051A 2045 39                      RTS              RETURN TO MAIN PGM

00053                       *
00054                       * SUBROUTINE MULT
00055                       *   RES := RES * 10  DIGIT SHIFT LEFT
00056                       *

00058A 2046 C6     0A     A MULT    LDB    #10       SETUP BCD=10
00059A 2048 B6     2002   A         LDA    RES       READ RES UPPER BYTE
00060A 204B 3D                      MUL              EXEC RES:=RES*10 (UPPER)
00061A 204C F7     2002   A         STB    RES       SAVE RESULT
00062A 204F B6     2003   A         LDA    RES+1     READ LOWER RES
00063A 2052 C6     0A     A         LDB    #10       SET UP FOR *10
00064A 2054 3D                      MUL              EXEC (RES+1):=(RES+1)*10
00065A 2055 F7     2003   A         STB    RES+1     SAVE RESULT (LOWER)
00066A 2058 BB     2002   A         ADDA   RES       ADD PREVIOUS DATA
00067A 205B B7     2002   A         STA    RES       SAVE RESULT
00068A 205E 39                      RTS
```

(b) 再入型程序

```
00012            2000     A BCDBIN EQU    *
00013A 2000 34   56       A        PSHS   D,X,U
00014A 2002 C6   04       A        LDB    #4        DIGIT COUNT
00015A 2004 8E   0000     A        LDX    #0        RESULT INITIALIZE
00016A 2007 34   14       A        PSHS   B,X       SAVE ONTO STACK
00017A 2009 C6   10       A LOOP   LDB    #$10
00018A 200B EE   63       A        LDU    3,S       READ INPUT DATA FROM STACK
00019A 200D 8D   19   2028         BSR    MULTBU    B,U=(U)*(B)
00020A 200F EF   63       A        STU    3,S       UPDATA INPUT DATA
00021A 2011 4F                     CLRA
00022A 2012 E3   61       A        ADDD   1,S       ADD TO PREVIOUS RESULT
00023A 2014 6A   E4       A        DEC    ,S        DECREMENT DIGIT COUNT
00024A 2016 27   0A   2022         BEQ    CMPLT     COMPLETED ?
00025A 2018 1F   03       A        TFR    D,U
00026A 201A C6   0A       A        LDB    #10
00027A 201C 8D   0A   2028         BSR    MULTBU    B,U=(U)*(B)
00028A 201E EF   61       A        STU    1,S       SAVE RESULT
00029A 2020 20   E7   2009         BRA    LOOP

00031A 2022 ED   63       A CMPLT  STD    3,S       OVERWRITE RESULT
00032A 2024 32   63       A        LEAS   3,S       RESTORE STACK
00033A 2026 35   D6       A        PULS   D,X,U,PC

00035                       *
00036                       * SUBROUTINE B,U=(U)*(B)
00037                       *

00039A 2028 32   7D       A MULTBU LEAS   -3,S      RESERVE LOCAL VARIABLE AREA
00040A 202A 34   44       A        PSHS   B,U       SAVE INPUT ONTO STACK
00041A 202C A6   61       A        LDA    1,S       READ U-HIGH BYTE
00042A 202E E6   E4       A        LDB    ,S        READ (B)
00043A 2030 3D                     MUL              D=(B)*(U-HIGH)
00044A 2031 ED   63       A        STD    3,S       SAVE ONTO STACK
00045A 2033 A6   62       A        LDA    2,S       READ U-LOW BYTE
00046A 2035 E6   E4       A        LDB    ,S        READ (B)
00047A 2037 3D                     MUL
00048A 2038 AB   64       A        ADDA   4,S
00049A 203A ED   64       A        STD    4,S
00050A 203C 32   63       A        LEAS   3,S       DELETE LOCAL VARIABLE
00051A 203E 35   C4       A        PULS   B,U,PC
```

3.4.4 递归型程序设计

递归程序是能调用自己本身的一种程序（又称递归调用程序）。众所周知，在Pascal语言和语言处理程序（LISP）之中，广泛使用着递归程序，最近在汇编语言中也在频繁地应用。特别是对大范围的变量处理、文章的分析、翻译等，有一连串数据接连出现，而且在数据块结束，尚需做出数据累计时，使用递归型程序设计是非常方便的。

许多情况下，递归子程序在执行处理的中途的某个时间，就需要调用自己本身。这时需要把中间结果原封不动地保留到该子程序的工作区，当其重新调用出自己本身的时候，一定不能破坏原来那些情况。也就是说，这一点和再入程序有相同的要求。再入程序有不用堆栈实现的方法，而递归程序只有使用堆栈的方法才能实现。

递归程序的典型例子是求n的阶乘的处理。n的阶乘n！，可以写为n！：=(n－1)！*n。因为(n－1)！可以表示为n！的表达式，所以在n！式中应求出n减1的数值。

表3.31为求阶乘的递归型子程序。

表3.31 递归程序（求阶乘的子程序）

```
00011P 0000 A6   C4        A FACT   LDA    ,U       FETCH N TO ACC A
00012P 0002 27   0B   000F          BEQ    FEND     IF N = 0 THEN RETURN
00013P 0004 4A                      DECA
00014P 0005 36   02        A        PSHU   A        N=N-1 SAVE ONTO U-STACK
00015P 0007 8D   F7   0000          BSR    FACT     CALL FACT BY RECURSIVE
00016P 0009 37   06        A        PULU   D        N AND N-1 FROM U-STACK
00017P 000B 3D                      MUL             ACC D (A:B) := N * (N-1)
00018                      * SAVE LEAST 8-BIT DATA ONTO U-STACK
00019P 000C 36   04        A        PSHU   B
00020P 000E 39                      RTS

00022P 000F 6C   C4        A FEND   INC    ,U
00023P 0011 39                      RTS

00025                      * FACTORIAL MAIN ROUTINE

00027P 0012 86   05        A MAIN   LDA    #N
00028P 0014 36   02        A        PSHU   A        SAVE IT ONTO U-STACK
00029P 0016 8D   E8   0000          BSR    FACT
00030P 0018 37   02        A        PULU   A        GET ANSWER FROM U-STACK
00031P 001A 12                      NOP
00032P 001B 3F                      SWI             BREAK POINT / END OF PGM

00034                      * SAMPLE RUN FOR N=5

00036            0005      A N      EQU    5
```

3.4.5 协同程序

协同程序的原始含意是把一个程序分离成一部分为输入/输出，另一部分为数据处理，目的是能够执行多道任务的要求。在一般情况下，输入/输出程序随着所用计算机的资源不而改变（由于构成的终端不同），因此，就需要把包含在数据处理本身之中大量的输入/输计同出装入模块进行改变。在分离输入/输出的过程中，应当注意使输入/输出程序适应于各个算机的要求。

在有几个任务都要调用的子程序中，当伴随有输入/输出操作时，如其中有某个子程序要求使用输入/输出程序，这时，要看由哪个任务来调用它们而有所区别。也即必须由主程序给出所要使用的输入/输出程序的起始地址。同时，也有的还需根据要求给出更多的参数。

为满足上述要求，在实现手段上，和前节讲过的程序设计技巧不同，可以有许多方法。里这选用在主程序中，把所需要的参数表示在应该调用的子程序指令之后的方法为例加以说明。

多道任务之例由于难于理解，所以仅以简单的数据块传送为例来进行说明，如表3.32所示。如果从协同程序返回时使用RTS指令，因为没有记忆返回参数部分，所以在协同程序内不能使用RTS指令；而应使用EXG指令，使程序计数器（PC）和其它的16位寄存器进行交换后返回主程序。在本程序例中，16位寄存器采用的是用户堆栈指示器。

表3.32　协同程序（数据块传送子程序）

```
00007                                  *
00008                                  * MAIN ROUTINE
00009                                  * EXEC EXAMPLE START  = $1000
00010                                  * DESTINATION ADDRESS = $8000
00011                                  * BLOCK SIZE          =   $80
00012                                  *

00014                1000      A START  EQU     $1000
00015                8000      A DESTN  EQU     $8000
00016                0080      A SIZE   EQU     $80

00018                0000      P MAIN   EQU     *
00019P 0000 8D       07   0009          BSR     MOVE
00020P 0002          1000      A        FDB     START
00021P 0004          8000      A        FDB     DESTN
00022P 0006          80        A        FCB     SIZE

00024P 0007 12                          NOP
00025P 0008 3F                          SWI              BREAK POINT / END OF RUN

00027                                  *
00028                                  * BLOCK TRANSFER CO-ROUTINE
00029                                  *

00031P 0009 EE   E1            A MOVE   LDU     ,S++     FETCH TRANSFER PARAMETER
00032                          *                       AND ADJUST SYSTEM STACK POINTER
00033P 000B AE   C1            A        LDX     ,U++     FETCH START ADDRESS
00034P 000D 10AE C1            A        LDY     ,U++     FETCH DESTINATION ADDRESS
00035P 0010 E6   C0            A        LDB     ,U+      FETCH BLOCK SIZE
00036                          *                       USER STACK = EXIT ADDRESS

00038                          * TRANSFER ROUTINE / LOOP
00039P 0012 A6   80            A MOVE1  LDA     ,X+      FETCH ORGINAL DATA
00040P 0014 A7   A0            A        STA     ,Y+      TRANSFER DATA
00041P 0016 5A                          DECB             LOOP IF SIZE > 0
00042P 0017 26   F9   0012              BNE     MOVE1

00044P 0019 1E   35            A        EXG     U,PC     RETURN TO MAIN PGM
```

3.4.6　全变量和局部变量(堆栈区的作业)

6809中有系统堆栈指示器S和用户堆栈指示器U。系统堆栈指示器的主要用途是：当使用子程序或发生中断时，把MPU中寄存器内容保留到堆栈之中，或者从堆栈返回时进行管理之用。因为这些处理过程都用MPU的硬件进行处理，所以，程序设计人员不必要特别考虑。但在软件方面，程序设计人员应该了解的是：对系统堆栈指示器进行操作的内容，即在各个子程序中MPU寄存器的保留和返回的内容。

6809的系统堆栈指示器，因为和其它16位寄存器一样，可作为变址寻址的指示寄存器使用，所以也可对系统堆栈内的数据直接进行运算和数据传送等处理操作。

堆栈的这些数据处理，只是各种子程序使用的作业，即适用于局部变量，因而存储器的使用效率提高，特别是可以不用高速暂存器。由于这些特点，所以易于实现子程序标准化（模块化或宏化），这样就会大幅度地改进以后的软件开发能力。

局部变量设置在系统堆栈之内，由其它子程序传送参量，如果使用用户堆栈作为全变量

传送，那么在处理局部变量和全变量两方面的子程序中，对数据的访问也就不需要使用绝对地址。进一步讲，如果程序本身也用位置独立型程序设计技巧编写，那么把那些标准化的程序模块作为宏指令时，就有可能为更多的程序设计人员所利用。而这些程序可以固化在ROM之中作为子程序使用；同时，使用存储器管理单元进行管理，这样就可以把它们放在任意的位置上。

6809中这样做的程序模块化产品有MC6838浮点运算ROM。该模块由于是位置独立型程序因而可以放在系统内的空闲区域之中，为其它程序任意使用。

关于用户堆栈指示器的用途问题，一部分内容在1.2.2节中已做过些说明。作为不同的用途，可以再举出它易于进行全变量处理的例子。在此所说的全变量，实质是局部变量的意思。根本上说，只是在某个子程序内所使用的作业称局部变量，除此之外，都可以看作全变量。

下面将以全变量和局部变量的具体处理方法为例进行说明。

1. 全变量处理

全变量一般是指几个子程序可以共同使用的作业。

6809中全变量处理是使用用户堆栈指示器U进行处理的。如果采用系统堆栈指示器S作为指示全变量的指示器，那么在每次调用子程序时，从子程序返回的地址将被保留在系统堆栈区，而没有进入全变量区中。这样做的结果，不能使子程序模块化和结构化。但如果使用用户堆栈指示器，就不会发生这种问题。现在对处理全变量的程序例加以说明。

表3.33为全变量数据处理之例。

在该程序中，所用变量的起始地址、变量的大小以至执行的种类等这些往子程序传送的参数，将称为内参数（In-line parameter）传送。这种传送只是在MPU的内部寄存器不能传送所有信息时，传送内参数是方便的。

子程序BLOCK的开始指令为LDU，S，它是把全变量的起始地址（调用BLOCK的BSR指令的下一条地址，即为原来用RTS指令返回的地址）存在用户堆栈指示器的指令，是读取全变量的前处理操作。

从LDX，U++到LDD，U++这段程序是接受来自主程序的参量，传送到MPU的内部寄存器之中。执行过LDD，U++时，用户堆栈指示器的内容，由于被更新为全变量的下一行，即NOP的地址，所以系统堆栈内的返回地址用STU，S预先设置好。

从LEAU <TABLE，PCR到STA OPCODE，PCR这段程序是用程序计数器相对寻址方式取入由FLAG所规定的指令操作码，即暂定为指令ADDA。在该程序中，程序本身由于要修改自己的程序，所以只能在RAM中执行；而且只是增加表格的内容，就可以进行各种各样的处理。在做成ROM化程序时，只要把EXEC子程序用数据块传送程序移到RAM之中即可执行。

调用子程序时，对于和主程序之间的变量交接，一般有两种方法，一种是把所有的变量都存在寄存器之中，但使用变址寻址方式；另一种是通过用户堆栈进行的方法。前者也称为内参数传送。

关于内参数传送问题

编模块化子程序时，需要把参数交接所要求的数据区用绝对地址给出来。除去外围器件I/O等地址以外，对于完全不使用绝对地址的位置独立型方法或者完全不用高速暂存器RAM

的再入程序技巧来说，这一点是不方便的。因此，在6809中采用变址寻址方式的内参数传送方法是较为理想的。所谓内参数传送，就是在分支转移到子程序的指令下边设置参数形式的交接方法。因此在被调用的程序内，仅要求所设置的数据字节数能达到返回地址的数值即可。表3.28中的程序采用的就是这种方法。优点是在子程序内部可以任意地使用所有的寄存器。6809所使用的编译程序，基本上讲大多数使用的是内参数传送。

2. 局部变量的处理

局部变量指的是只在各个子程序之内使用的作业区。如果某个子程序的处理全部结束，那么该子程序的局部变量就可以完全不要。如果考虑到这种性质，又要确保在系统堆栈内的数据区，怎样做是合理的，方法不难找到。

以局部变量来说，确保其在系统堆栈区域内的方法有两个。第一个方法，在进行返回的同时，不管数据是否被破坏，如果MPU本身有寄存器可作为参数输入的寄存器使用，那么就可以用PSHS指令保留到系统堆栈区，同时利用系统堆栈指示器，按变址寻址方式来处理数据。第二个方法，用LEAS －n，S指令，在系统堆栈内确保n字节区域，而在子程序返回之前，用LEAS n，S指令来作废被确保的区域。并且需把局部变量的总字节数确保在系统堆栈区的上面，用常数偏值的变址寻址方式处理数据。

局部变量的确保和作废的一般情况：

```
SUBR   PSHS   REG        保留寄存器
       LEAS   -n, S      确保局部变量区
         ⋮
       LEAS   n, S       作废局部变量区
       PULS   REG,PC     恢复寄存器和子程序返回
```

在STA －1，S或INC －2，S等指令中，偏值为负数时使用了系统堆栈内的作业区，如果发生了中断，这时就会破坏使用中的作业区。为了防止这种情况，在开始用LEAS －n，S这条指令时，要把系统堆栈指示器的数值设置到所用作业区的下限。即使在Pascal高级语言中，所用的局部变量、局部配置都要保证作业区在系统堆栈之内，才能写出合理的程序。

表3.33 全变量数据处理

```
00008                          * MAIN ROUTINE

00010              2000     A EXAM    EQU     *
00011A 2000 8D     08   200A          BSR     BLOCK
00012A 2002        4000     A         FDB     ARGUM1    GLOBAL ARGUMENT 1
00013A 2004        5000     A         FDB     ARGUM2    GLOBAL ARGUMENT 2

00015                          * RESULT STORED IN ARGUMENT2 REGION

00017A 2006        01       A         FCB     FLAG
00018A 2007        64       A         FCB     SIZE      ARGUMENT SIZE

00020                          * GLOBAL DATA DEFINITION

00022              4000     A ARGUM1  EQU     $4000
00023              5000     A ARGUM2  EQU     $5000
```

```
00024               0064    A SIZE    EQU     100
00025               0001    A FLAG    EQU     1

00027                         * FLAG    :     0  ADDITION
00028                         *               1  SUBTRACT
00029                         *               2  EXCLUSIVE-OR
00031                         * MAIN PROGRAM CONTINUE

00033A 2008 12                          NOP
00034                         * END OF RUN

00036A 2009 3F                          SWI

00038                         * ARRAY DATA PROCESS SUBROUTINE

00040               200A    A BLOCK   EQU     *

00042                         * INITIALIZE GLOBAL DATA POINTER

00044A 200A EE     E4       A           LDU     ,S

00046                         * FETCH ARGUMENT1 ADDRESS

00048A 200C AE     C1       A           LDX     ,U++

00050                         * FETCH ARGUMENT2 ADDRESS

00052A 200E 10AE C1         A           LDY     ,U++

00054                         * FETCH BLOCK SIZE AND FLAG

00056A 2011 EC     C1       A           LDD     ,U++

00058                         * UPDATE SUBROUTINE RETURN ADDRESS
00059A 2013 EF     E4       A           STU     ,S

00061                         * FETCH EXECUTIVE OPCODE

00063                         *         FETCH TABLE ADDRESS TOP

00065A 2015 33     8C 10                LEAU    <TABLE,PCR

00067                         * A=FLAG U=TABLE ADDRESS TOP

00069A 2018 A6     C6       A           LDA     A,U

00071                         *                      THEN  A := (U+A)

00073A 201A A7     8D 0002              STA     OPCODE,PCR SET OPCODE

00075                         * EXECUTE PROGRAM

00077A 201E A6     80       A EXEC      LDA     ,X+        FETCH ARGUMENT1

00079A 2020 AB     A4       A OPCODE    ADDA    ,Y         EXECUTE FUNCTION
00080A 2022 A7     A0       A           STA     ,Y+        STORE RESULT
00081A 2024 5A                          DECB               SIZE := SIZE-1
00082A 2025 26     F7    201E           BNE     EXEC       REPEAT IF SIZE > 0
00083A 2027 39                          RTS

00085                         * OPCODE TABLE

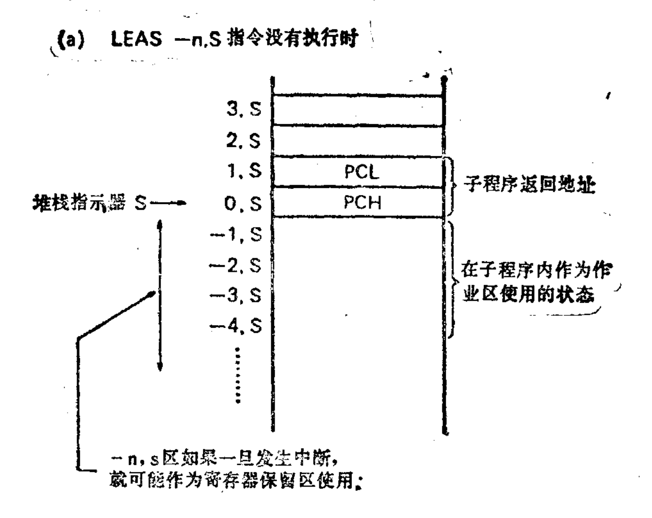
00087               2028    A TABLE     EQU     *

00089                         * ADDA INDEXED ADDRESSING MODE
00090A 2028        AB       A           FCB     $AB

00092                         * SUBA INDEXED ADDRESSING MODE
00093A 2029        A0       A           FCB     $A0

00095                         * EORA INDEXED ADDRESSING MODE
00096A 202A        A8       A           FCB     $A8
```

图3.31是在中断条件下堆栈区的使用方法。



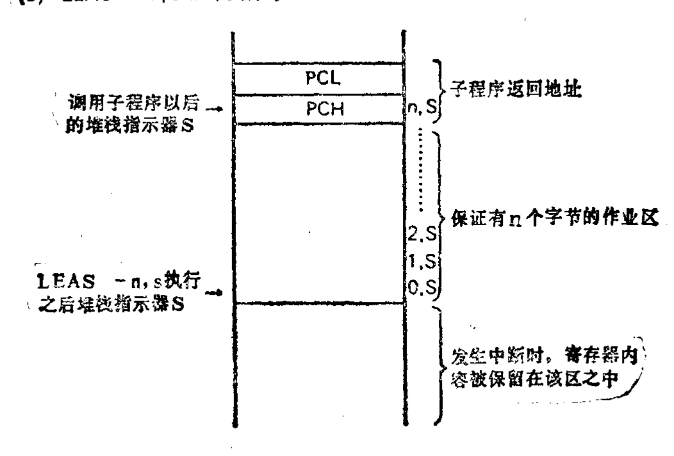


图3.31 考虑堆栈条件下堆栈区的使用方法

表3.34是从二进制数变换为BCD数，并由BCD数变换为ASCII编码的子程序。

从BINBCD开始的子程序的行号为21～25，这一段程序是把内参数所传送的全变量传送到局部变量区的预处理程序。

输入/输出数据地址放到X、Y寄存器之中，使它们作为 BNBCD的进入地址。

从BNBCD到BNBCD1 是确保局部变量和数据设置的程序。

BNBCD 2 是用常数去除输入数据的程序，其商放在系统堆栈指示器所给地址的下个地址(1 ,S)。BNBCD 3 为常数位的更新和使处于1，S中的商数变换为 ASCII 代码，并在OUTPUT地址处输出。到位数计数器 0，S为 0 之前，则重复从BNBCD 1 的循环。位数计数器为 0 时，则把局部变量区中的 4 个字节作废，并恢复X、Y寄存器内容，从而按照恢复好的程序计数器内容返回到主程序上。

表3.34　局部变量的处理/二进制数变换为ASCII

```
00019                            * ENTRY IN-LINE PARAMETER TRANSFORM

00021              2000      A BINBCD EQU    *
00022A 2000 EE     E4        A        LDU    ,S       FETCH GLOBAL DATA ADDRESS
00023A 2002 AE     C1        A        LDX    ,U++     FETCH BINARY DATA ADDRESS
00024A 2004 10AE   C1        A        LDY    ,U++     FETCH ASCII DATA ADDRESS
00025A 2007 EF     E4        A        STU    ,S       UPDATE RETURN ADDRESS

00027                            * REGISTER DATA TRANSFORM ENTRY

00029              2009      A BNBCD  EQU    *
00030A 2009 34     30        A        PSHS   X,Y      SAVE GLOBAL DATA ONTO S-STACK
00031A 200B 32     7C        A        LEAS   -4,S     RESERVE 4 BYTES LOCAL VARIABLE
00032A 200D 33     8C 23              LEAU   <KCONST,PCR FETCH KCONST TABLE ADDRESS
00033A 2010 C6     05        A        LDB    #5       DIGIT COUNT SET
00034                            * SAVE DIGIT COUNT ONTO LOACAL VARIABLE
00035A 2012 E7     E4        A        STB    ,S
00036A 2014 EC     F8 04     A        LDD    [4,S]    FETCH 16-BIT BINARY DATA

00038A 2017 6F     61        A BNBCD1 CLR    1,S      DIVIDE RESULT = 0
00039A 2019 A3     C4        A BNBCD2 SUBD   ,U       D:=D-(U)
00040A 201B 25     04   2021          BCS    BNBCD3
00041A 201D 6C     61        A        INC    1,S      INCREMENT DIVIDE RESULT
00042A 201F 20     F8   2019          BRA    BNBCD2   CONTINUE UNTIL OVERFLOW OCCUR

00044A 2021 E3     C1        A BNBCD3 ADDD   ,U++     ADJUST TEMPORARY DATA IN ACC D
00045A 2023 ED     62        A        STD    2,S      SAVE TEMPORARY DATA
00046A 2025 A6     61        A        LDA    1,S      RESTORE DIVIDE DATA
00047A 2027 8B     30        A        ADDA   #'0      CONVERT TO ASCII
00048A 2029 A7     A0        A        STA    ,Y+      SAVE ASCII DATA
00049A 202B 6A     E4        A        DEC    ,S       DECREMENT DIGIT COUNT
00050A 202D 26     E8   2017          BNE    BNBCD1   COMPLETE ?
00051A 202F 32     64        A        LEAS   4,S      DELETE LOCAL VARIABLE AREA
00052A 2031 35     B0        A        PULS   X,Y,PC   RESTORE REG AND RETURN

00054                            * CONSTANT DATA

00056A 2033        2710      A KCONST FDB    10000
00057A 2035        03E8      A        FDB    1000
00058A 2037        0064      A        FDB    100
00059A 2039        000A      A        FDB    10
00060A 203B        0001      A        FDB    1
```

3.4.7　软件中断的应用

如前所述，6809设有SWI 1、SWI 2、SWI 3三个软件中断指令。6809的SWI 1和6800的SWI相同，它使条件码寄存器的高2位（E、F）标志位置1。但SWI 2和SWI 3都不能使E、F标志位发生变化。

软件中断指令的第一个特点是指令本身只有操作码，没有操作数。第二个特点是由软件中断所决定的向量地址的内容可以跳到64K字节的任意空间。在执行软件中断时，由于全部寄存器都保留在系统堆栈之内，所以可以利用主程序内寄存器中的内容；而且新的作业也可以利用这些寄存器，在用过之后，全部寄存器的内容还可以进行恢复。

在软条中断处理程序内，也可以利用再入方法，接连不断地进行软件中断处理。各个软件中断都是独立存在的，所以在完成大型任务时，需要把程序设计成结构化的或是模块化的

程序。一般情况，往往把软件中断作为执行宏指令使用。图3.32是使用SWI作宏指令之例。在该例中，宏指令可有128种。在中断程序中，为了用运算方法求出表格所在地点，可以使用程序计数器相对寻址（LEAX MACRO, PCR），累加器为偏值的间接变址寻址方式（JMP 〔B, X〕）。

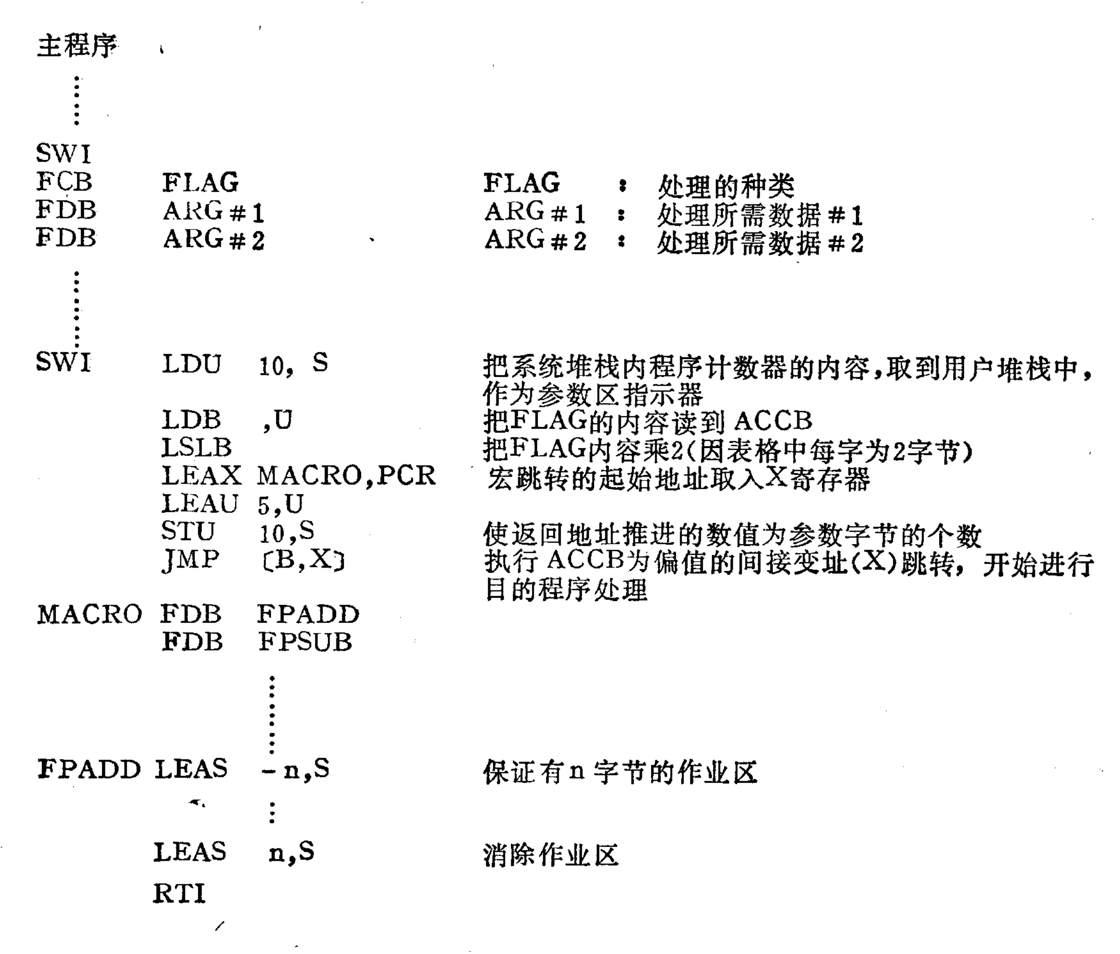


图3.32 使用SWI作宏指令

宏指令调用的实例如表3.35 (a) 中所示的程序。

有关宏汇编程序的表示方法，请参考有关宏定义资料。不过这些程序都可做成再入型、位置独立型或递归型程序。

使用一字节内参数传送和SWI指令所设计的宏指令，这时操作数会增加一个字节的机器字。因此，对主程序而言，其前提条件是处理软件中断的操作系统应该完备。但各个操作的地址，不必要向设计主程序的程序人员公开。而需要公开的只是给出操作数值和操作的关系表格。该表格之例如表3.35 (b) 所示。

使用汇编语言进行程序设计时，和高级语言一样，多数程序设计人员还要划分为宏单位，设计主程序时也用宏指令编写，这样可以提高效率。

表3.35 宏调用和宏表格之例

(a) 宏调用

```
00057                       * PRINT STRINGS TERMINATED BY EOT ($4)

00059                       * PDATA CR/LF BEFORE STRINGS
00060                       * PDATA1             STRINGS ONLY

00062A 0020                 PDATA  XCALL  .PCRLF
00063A 0022 AE   44       A PDATA1 LDX    4,U      FETCH STRING TOP ADDRES
00064A 0024 A6   80       A PDATA2 LDA    ,X+      READ PRINT DATA
00065A 0026 81   04       A        CMPA   #$4      PRINT DATA = EOT ?
00066A 0028 27   06   0030         BEQ    PDATA3   RETURN IF EOT
00067A 002A                        XCALL  .OUTCH   OUTPUT ONE CHARACTER
00068A 002C AF   44       A        STX    4,U      RESTORE CHARACTER POINTER
00069A 002E 20   F4   0024         BRA    PDATA2
00070A 0030 39              PDATA3 RTS
```

(b) 宏表格

```
* SYSTEM CALL (XCALL) USING SWI ($3F) WITH ONE BYTE INLINE CODE ($80-$9F)
XCALL MACR
 IFEQ NARG-1
 SWI
 FCB  ¥0!+%10000000
 ENDC
 IFNE NARG-1
 FAIL *
 ENDC
 ENDM
 SPC 2
SEQ MACR
 IFNE NARG
¥0 EQU *
 ORG *+1
 ENDC
 ENDM
 SPC 2
 ORG $0
 SPC 2
 SEQ .INHEX INPUT ONE HEX CHARACTER / CONVERT HEX
 SEQ .BEGEN INPUT BEGIN AND END ADDRESS (INDEXED BY X-REG)
 SEQ .CBCDH CONVERT ASCII TO HEX
 SEQ .BCDBI CONVERT ASCII TO BCD
 SEQ .CHEXL CONVERT UPPER 4 BIT DATA TO ASCII
 SEQ .CHEXR CONVERT LOWER 4 BIT DATA TO ASCII
 SEQ .INADD INPUT ADDRESS STORED IN X-REG INDEXED ADDRESS
 SEQ .INCH  INPUT 8-BIT ASCII DATA FROM ACIA
 SEQ .INCHN INPUT 7-BIT ASCII DATA FROM ACIA BIT 7=0
 SEQ .OUTCH OUTPUT ONE CHARACTER FROM ACC A
 SEQ .OUT2H OUTPUT 2 HEX AND SPACE
 SEQ .OUT4H OUTPUT 4 HEX AND SPACE
 SEQ .PCRLF PRINT CARRIER RETURN AND LINE FEED
```

第四章　6809的接口、系统和应用

4.1　6809的接口

4.1.1　基本输入/输出

在大多数第三代微处理器（如6809）中，微处理器可以直接同MPU芯片的数据总线接口，也可以通过专门器件或者为某种目的而设计的特殊芯片接口。将来的微处理器，外围器件的许多功能将做在微处理器芯片之内。但不管哪种情况，这种接口都是用来组成微处理器输入/输出的结构。微处理器的输入/输出端可以是单线、多线、并行或串行结构。在本节将介绍一些很简单的输入/输出接口，有单线和多线的两种。这些接口都是常用的，所用器件有的不用时钟，有的可用6809时钟同步。

1. 基本输出形式

单线输入/输出是微处理器接口的最基本的形式。图4.1给出了一种极简单的单线输出的线路。该线路采用一个7474的D型触发器作成。当在7474的时钟输入端加入时钟信号时，该位作为锁存器就可以把6809数据总线D0的状态保存起来。时钟信号输入端，由A0、A1、A15、E和R/$\overline{W}$地址和信号线经过'与非'门进行驱动。例如可以使用一条简单的指令STA $8001来输出累加器A的内容。注意，指令STA $8005、STA $8009……都可以输出累加器A的内容。

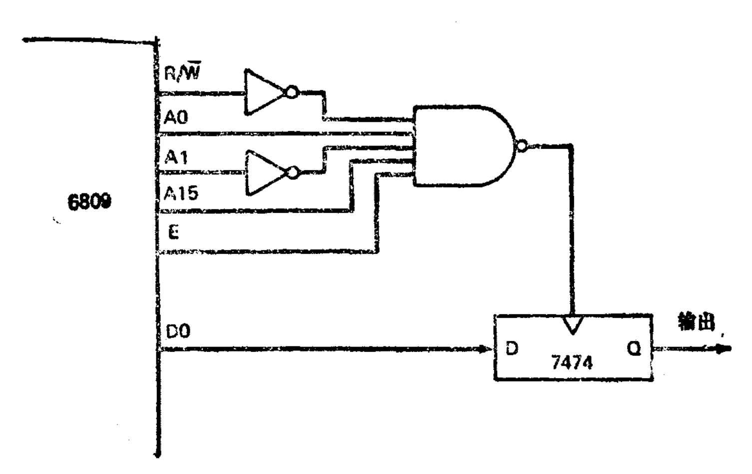


图4.1　单线输出端结构

这种输出可以很方便地作为某些装置的控制信号使用。该装置在触发器Q端读出0状态时，向某一特定方向动作；读出为1状态时，则反方向动作。显然，这种单线输出端在任意时间只能使装置本身选择两种状态中的一种状态。该信号在MPU的地址总线和E信号形成之后才会有效，并在地址总线状态稳定后，E的下降沿才能选通数据线信号。

单线输出产生的脉冲可正可负，图中给出的线路产生正脉冲。

根据图4.1所给线路，使用以下程序可以做成正脉冲产生器。

```
     LDB    # 1      产生脉冲
     STB    $8001    给出‘1’
     LDA    #$10     延迟数值
UP   DECA
     BNE    UP
     COMB
     STB    $8001    给出‘0’
```

2. 锁存输入基本形式

当某设备的一条输出线上可以给出数据，但通常并不保持该数据时，这时要求截获（锁存）该数据，其线路如图4.2所示。从线路可见，锁存器件的输出端接到三态缓冲驱动器，锁存器使用E脉冲的上升沿来截获瞬间的输入信号。同时也使三态器件启动工作。这时，该输入信号读进6809的 D_0 端，可以使用指令LDA　$8002（或$8006、$800A……）。这种方法满足输入端的最低要求（即经三态门的译码和定时），同时还具有锁存功能。

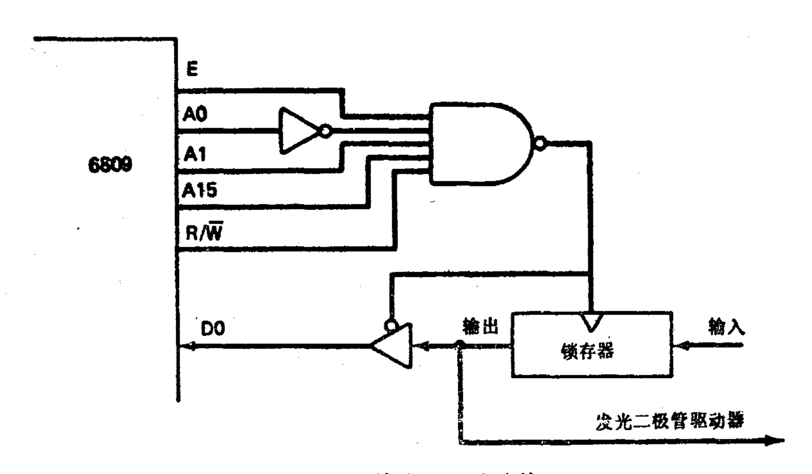


图4.2　单线输入端结构

如果接收输入端为多条并行信号，这时可以单线输入改成多线输入的结构。使用一个芯片的8位输入口如图4.3所示。而且这8个锁存器的输出并行接到6809的数据输入端上。这些锁存器由地址线、R/$\overline{W}$和E信号实现时钟信号同步。为了从这些锁存器中读出输入信号，可以采用指令LDA　$8002（或$8006、$800A，……）完成。

如果设备本身只要求电路尽量简单，那么使用起来也直截了当。但还有许多设备要求在设备和处理器之间产生某些回答响应或交接信号。这样耦合连接的结果就可以通知微处理器：设备已经为微处理器的工作准备就绪，或者相反还没有准备好。另外微处理器也可以通知外围设备；它的数据已经准备就绪。总之当其它单元需要肯定回答信号作为响应时，这种情况下，就会产生交接关系的要求，或称“握手”信号。这种控制信号的一种特殊解释可以称为通信协议，英语称作Protocol。意思也就是希望有一种符合一定规则、有秩序的交换。当把字节

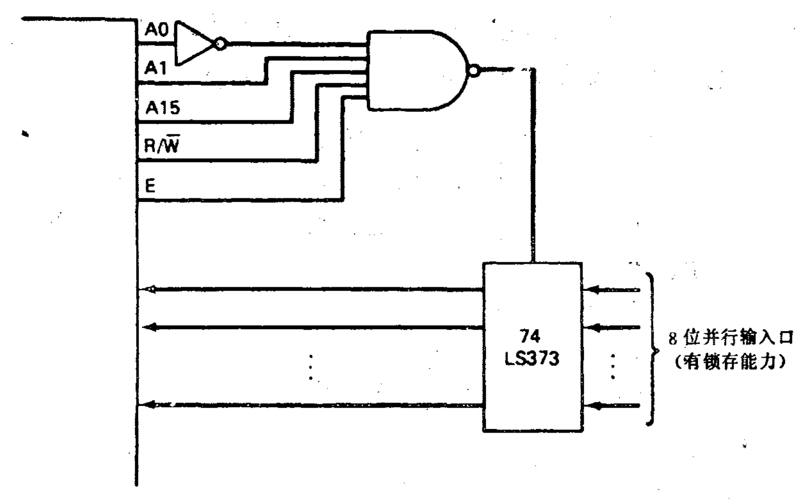


图4.3　8位并行输入口线路

和信息的格式以及信息的顺序都规定好之后，通信协议本身就会很严格、准确。微型计算机或微处理器通常都使用和它们本身系列相配套的外围器件，这些外围器件都设有标准的或者容易使用的工作交接的方法。

4.1.2　并行接口

当希望采用具有工作交接方式的并行接口时，68××系统通常使用MC6821并行接口芯片，有时简称PIA。并行的双输入/输出口实例如图4.4所示。PIA1作为双输出口，其A和B口都作8位输出。PIA2作为双输入口，其A和B口都作8位输入。例如该输入和输出口在实

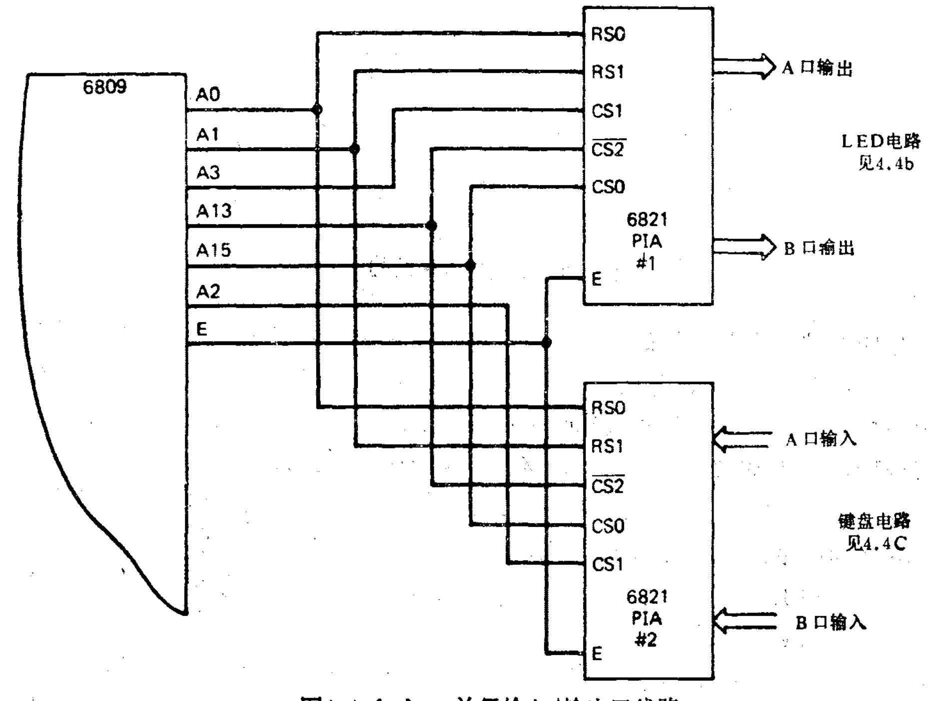


图4.4（a）　并行输入/输出口线路

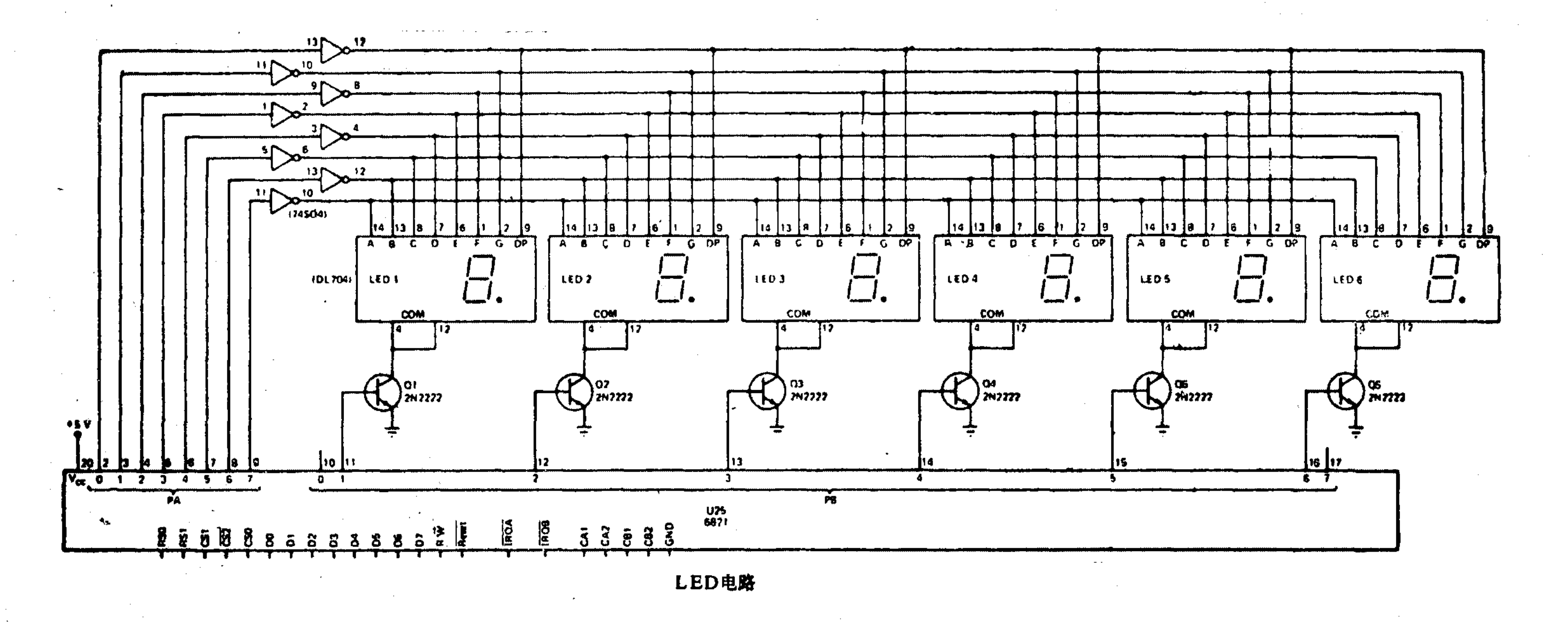


图4.4（b） LED电路结构

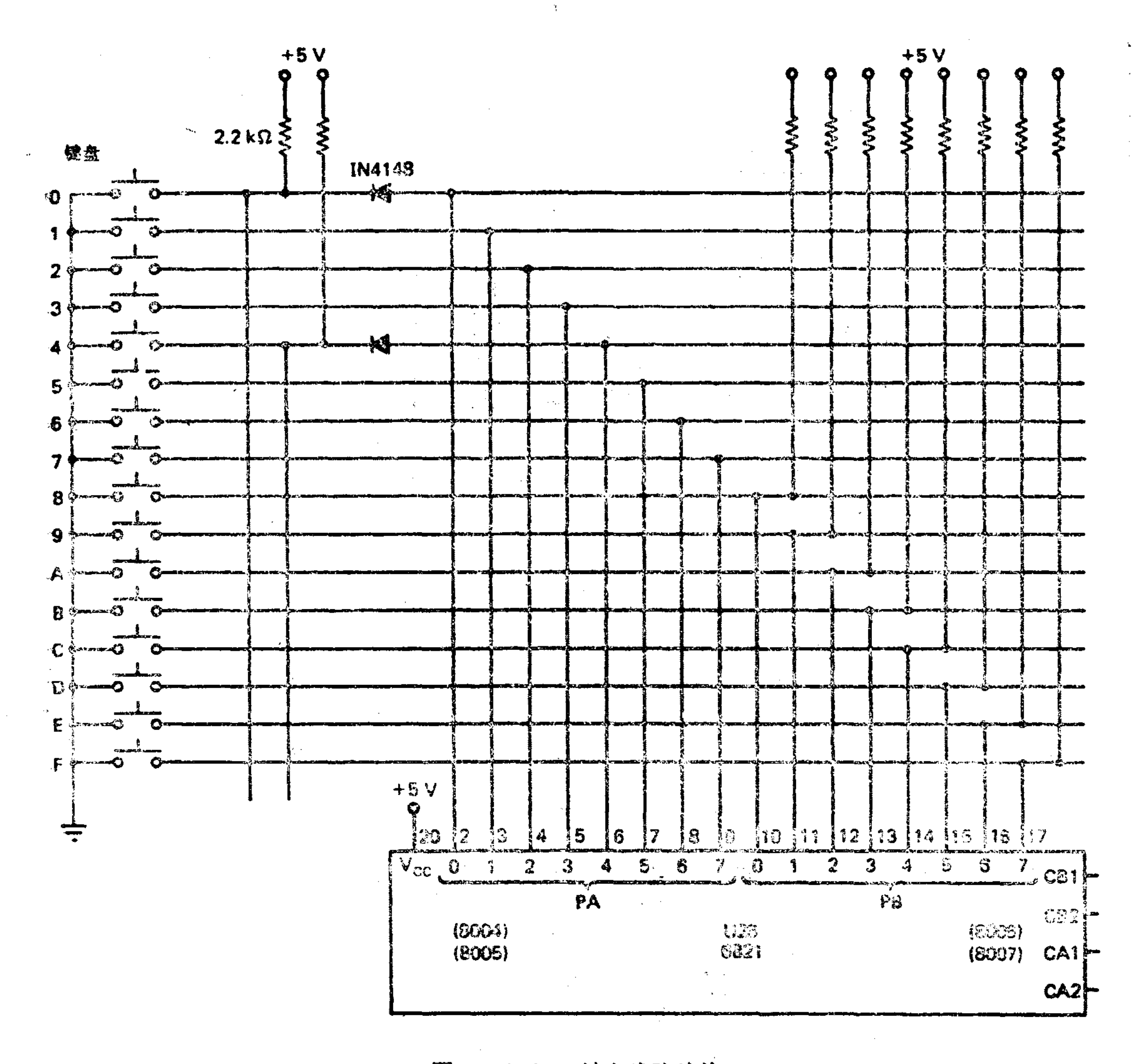


图4.4（c） 键盘线路结构

际应用中可分别接到键盘和光电二极管（LED）数字显示器上，给系统作外围设备使用，如图4.4（b）和图4.4（c）所示。其中PIA1的A和B输出口驱动显示电路；PIA2的 A 和B口分别接到键盘译码器的行和列输入端上。

1. 6821接口

6821是可程控的并行接口的I/O器件。每片6821都设有两个 8 位数据线接口，其中每个口都相应设有两条控制线。每个口是作为输入还是输出使用，其控制线是作为输入还是输出控制，这些功能的实现都可以由程序赋值，或重新定义。图4.5是 6821 作为输入口交接使用的线路结构。根据上列 PIA2 作为键盘输入口使用，有关寄存器的选择地址确定如下：A 口数据寄存器DRA地址为＄8004，A口控制寄存器CRA地址为＄8005，B 口数据寄存器 DRB地址为＄8006，B口控制寄存器 CRB 地址为 ＄8007。PIA1 作为输出口使用，B口数据寄存器DRB地址为＄800A，B口控制寄存器CRB地址为＄800B（有关6821的详细说明见参考文献资料〔1〕）。

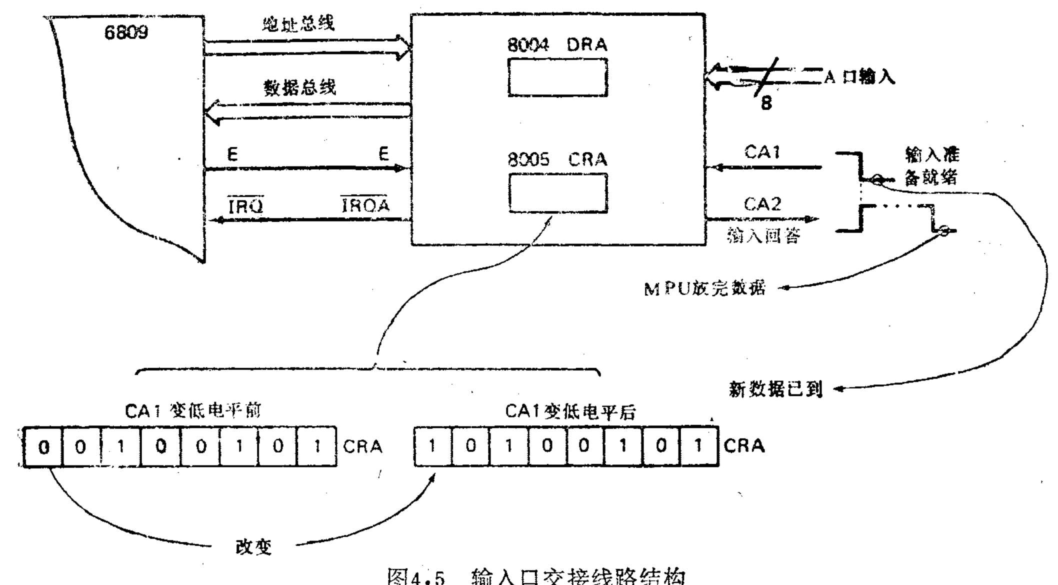


图4.5　输入口交接线路结构

在应用中选择PIA各寄存器时，主要是使用6821芯片上所设置的输入线RS_0、RS_1、CS_0 CS_1和$\overline{CS_2}$。这些输入线要适当接到地址总线的 A_0～A_{15} 相应的地址线上，并以此来译码。PIA芯片设有同步信号，在MPU适当的时间对数据进行锁存。当6821同6809系统相连时，要把6809的 E 信号接到6821的E端之上。凡是可编程控制的器件都需要做一些初始化器件状态的工作。6821 器件在系统 RESET 信号作用下，会自动地使6821中所有寄存器清零。如果需要改变A和B口的输入/输出方向以及控制方式，设计人员可以用程序改变数据方向寄存器的内容（DDRA和DDRB），或者改变控制寄存器CRA和CRB的内容。

一个完整的微处理器系统中的加电动作与 RESET 信号是同时产生的，所以有可能使模糊的定时信号加到6809相连的各种接口电路上，误以为是 RESET 的结果。为了防止这种问题，6809在设计中要考虑到这一点。当时6809和6821以及几乎其它所有68××外围器件进行清零（RESET）时，6809将是最后完成清零的器件，因此当 6809 设置它们的工作内容时，这些外围器件都将做好了工作准备。所以，不会使随机的数据或控制字偶然地锁存进可编程外围器件的寄存器之中。

（1）输入交接过程：

假设使用6809和在几个PIA中选一个作为输入口。同时，还假定器件本身希望通知MPU：数据已经到达。显然对该通信协议的信号可以由PIA中的CA1和CA2控制线很方便地实现。现在来考虑怎样选择图4.5中的信号格式以及相应地建立控制寄存器程序。设 A 口地址为 \$8004单元的是从某种设备来的 8 位输入口，A的控制CRA的地址单元为 \$8005。同时还假设6821中的IRQA线接到MPU的IRQ端。当该设备给出有效输入数据时，它就向PIA的 CA1线发出一个正到负沿的信号，这就是从设备发出来的“输入准备就绪”的信号。这时表明新的数据已经来到。当 CA1 变低电平时，CRA 的第 7 位置 1 。这时微处理器即被中断，并取出中断向量转入中断程序。在中断程序中的处理软件首先检查所有 PIA 器件中的CRA 寄存器，以便确定哪个PIA中CRA 的第 7 位为 1 。这时可以使用简单的分支转移指令 BMI 来测试第 7 位结果，因为第 7 位是一个机器字的符号位。

把数据读入6809的累加器 A 使用一条指令 LDA　$ 8004 即可完成。当 MPU 读出单元 $ 8004中的数据时，PIA将发出正向信号给CA2，通知外部设备，数据已由MPU取走。读出该数据寄存器内容之后，控制寄存器的第 7 位还要被清零，同时释放IRQ线。以上是6821作为输入口的一个有代表性的交接通信协议。当然还有其它方式也都是可行的。

下面根据图4.5的线路和工作过程来写出输入交接工作的程序。当经IRQA接受了中断以后，MPU就将执行中断处理程序，读出接到A口外设中的数据，该数据寄存器地址为 $ 8004。原来的ACCA内容先保存起来（在该程序之后恢复）。把数据装入ACCA，然后再把它存入RDATA单元。在离开中断之前，使I标志位清零。程序如下：

```
INTA  LCA     $8004    输入数据
      STA     RDATA    存储数据
      ANDCC   #$EF     ／#中断
      RTI
```

（2）输出交接过程

如果要求用交接通信协议建立一个 8 位输出口，与以上输入过程类似。如图 4.6 所示，设使用PIA器件的B口作为 8 位输出口使用，控制线CB1和 CB2 分别作为“输出请求”（设备需要其它数据），和“输出准备就绪”（MPU 已把数据装入数据寄存器）。当 CB1 变低电平时，该控制寄存器第 7 位将置 1 。当6809 经 IRQB 被中断时，中断处理软件应检查所有PIA控制寄存器中的第 7 位内容。6809的中断服务程序在响应中断处理时应把数据放在 $ 800A单元，这时可使用STA　$ 800A指令。因此数据从ACCA进入PIA的B口数据寄存器。

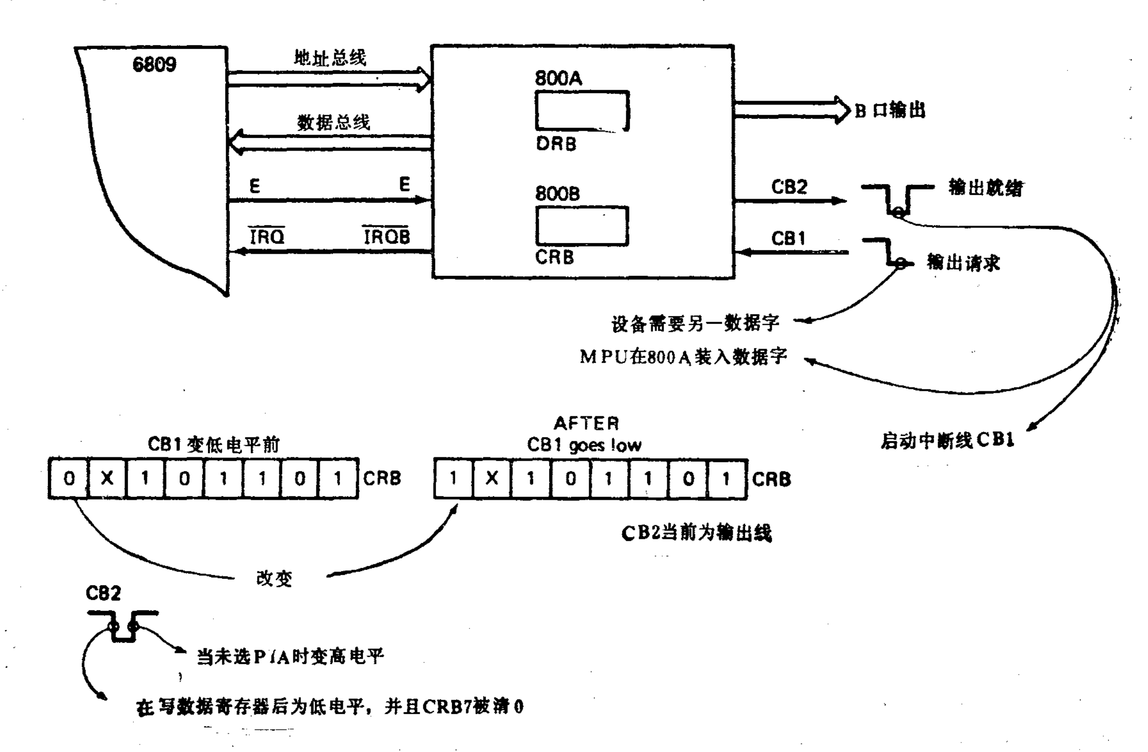


图4.6　输出口交接线路结构

同输入口交接协议一样，如果用程序适当地写入控制寄存器就会实现协议的要求，在$800A装入了数据之后，CB2将变为高电平状态。

上述图4.5和图4.6的交接协议的建立，要求对单元$8005和 $800B 的控制寄存器CRA和 CRB 分别用程序控制才可实现。所用的程序控制字在图中分别作了说明。因为6821是可程控的器件，所以可以改为输入控制线CA1、CB1和输出控制线CA2、CB2的内容解释。

下面给出输出交接工作的程序，程序中输出数据为25_{10}。经IRQB到MPU按受中断之后，为了使数据输出到PIA的B口，其程序如下，其中数据寄存器地址为$800A：

```
INTB  LDA     #$25
      STA     $800A    输出25
      ANDCC   #$EF     清零 I 屏蔽位
      RTI              返回
```

2. 模拟转换

众所周知，接收或产生模拟信号的重要的接口器件分别是模拟-数字转换（ADC）和数字-模拟转换（DAC）接口。使用 PIA作ADC 转换的线路如图4.7所示，6821（PIA）通过A口 8 位和B口 4 位采集由 12 位ADC 转换出来的数字量。模拟-数字转换器要求一个“启动转换”信号，从6821的CA2端产生正脉冲即可作为对 ADC 的启动转换命令。转换完成之后，ADC产生控制信号，表示“转换结束”或“数据有效”。该信号接到6821的CA1端。当该线为高电平时，表示数据在ADC中已经完全转换好，6821可以使用。

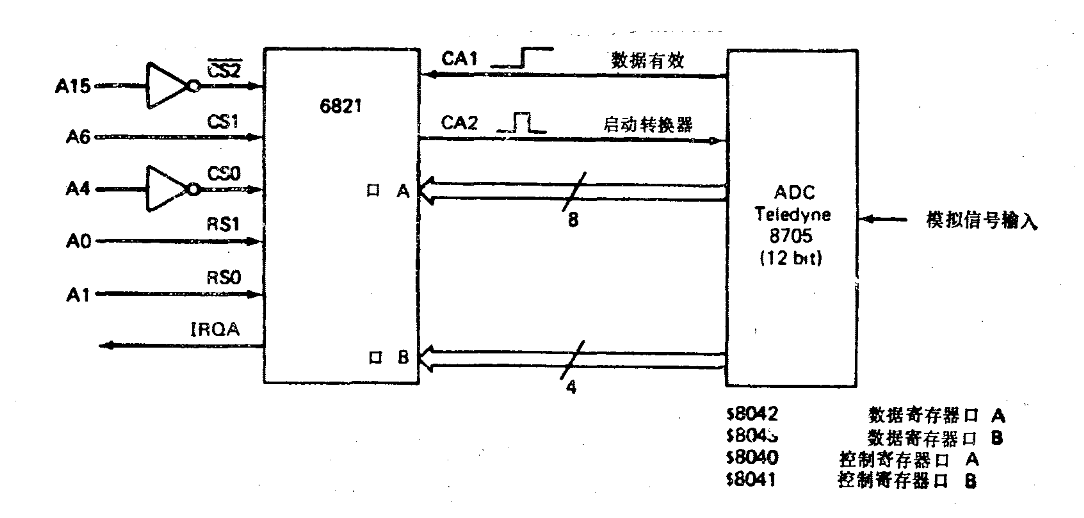


图4.7 12位模拟-数字转换接口

经6821的选片CS端和寄存器选择RS端，对A口和B口的控制寄存器和数据寄存器的特定地址单元设为$8040～$8043。图4.8所给的是6821直接连到数字-模拟转换器DAC的线路。

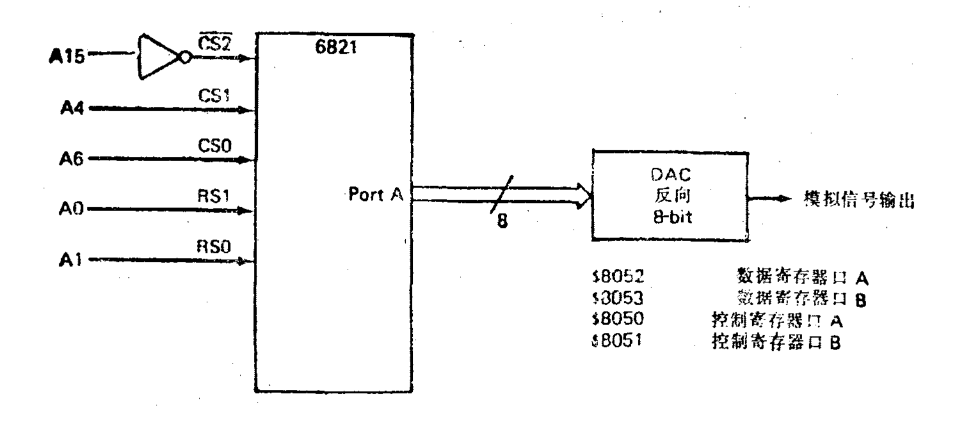


图4.3 8位数字-模拟转换接口

该DAC没有使用控制信号，而是自动地把8位数据转换为等效的模拟量。在DAC转换期间，需要保持数据的稳定性。6821的A口寄存器地址为$8050和$8052。经过地址总线使用选片端和寄存器选择线共同选定。

下面给出使用PIA作接口和ADC转换器把在－5～＋10V范围变化的模拟信号转换为数字量的程序。程序中所给的主程序是对图4.7的ADC电路和图4.8的DAC电路进行测试的程序。

表4.1是对ADC/DAC的测试程序。其过程实际上就是从ADC读出信号后就直接把它输出给DAC。后面的程序只是把12位转换数值中的高8位转换为模拟量。通常可以用这种程序作为ADC/DAC电路的诊断程序。程序中假定DAC是反方向的数字输入，即正数产生负的模拟信号。为了正确地使用该程序，在模拟信号送到ADC输入端之前需将其反向，一般加一个简单的运算放大器即可。模拟信号经过运算放大器后，在程序末尾设有进行比较的子程序以便确定数字量是否等效于送到DAC的数值，只要每次两个数值不等时，局部变量COUNT即被加1。如果ADC和DAC正常工作，COUNT变量数值一定为零。

在结束本节讨论之前，再说明一下在微型计算机系统和外设之间建立通信协议（约定）的几个条件，或者说是相互取得交接一致的要求。通常要求协议具有以下能力：

（1）外部设备对某些任务或服务的开始请求；

（2）被请求的系统对该请求做出肯定回答；

（3）外部设备接着做完服务；

（4）最后，外部设备在完成所接受的服务后做出肯定回答。

虽然每种硬件都可能大不一样，但这些通信约定的原则一定要按步就班地做到才行。实际中每一步骤都要用到性质不同的控制信号和肯定回答信号（或称确认信号）。这些信号通常称为："输入就绪"和"输入确认"，"输出就绪"和"输出请求"等。

表4.1 ADC/DAC线路测试程序

```
* THIS MODULE TESTS AN ADC/DAC BOARD FOR FAILURE THAT MAY
* OCCUR IN EITHER THE ADC OR DAC.  THE MODULE, TEST, CONSISTS
* OF A SET OF JUMPS TO SOME FIVE LOWER-LEVEL SUBROUTINES
* WHICH FIRST INITIALIZE THE PIA INTERFACES, THEN GENERATE
* A SAWTOOTH SIGNAL, STORING INTO THE DAC, CHECKING THE
* ADC SIGNAL TO VERIFY THAT THE OUTPUT SIGNAL CONVERTED
* BACK TO ITS DIGITAL ANALOG IS EQUIVALENT.  ANY ERRORS
* CAUSE LOCATION, /COUNT/, TO CONTAIN A NONZERO VALUE.  THE
* ROUTINE, TEST, CALLS SUBROUTINES PINIT, GINIT, DAC, ADC,
* AND CHCK.
*
*
* PINIT INITIALIZES DAC OUTPUT PORT A
* GINIT INITIALIZES ADC INPUT PORTS A, & B
* DAC   OUTPUTS    A WORD TO DAC VIA PORT A
* ADC   INPUTS     A WORD FROM 12 BIT ADC VIA PORT
                   A ( MSB ) AND PORT B ( LSB )
* CHCK COMPARES THE ADC DIGITAL VALUE WITH THE ORIGINAL
*       DAC VALUE, INCREMENTING /COUNT/ IF NO MATCH. ELSE
*       /COUNT/ IS ZERO.

* SINCE DAC IS INVERTING, AN ANALOG CIRCUIT MUST BE USED TO
* INVERT THE SIGNAL BACK TO ITS ORIGINAL POLARITY BEFORE
* ADC

* STACK PICTURE ON ENTRY AND EXIT

*       U+0        OLD STACK MARK
*       U-1        DAC OUTPUT VALUE
*       U-2        ADC INPUT VALUE
*       U-3        ERROR COUNT
*
* CALLING ROUTINE
TEST    PSHS       U,B,A,CCR
        TFR        S,U
        LEAS       -3,U
        CLR        VALOUT,U
        CLR        COUNT,U
        LBSR       PINIT
        LBSR       GINIT
LOOP    LBSR       DAC              OUTPUT TRIAL VALUE
        LBSR       ADC              RETRIEVE ANALOG VALUE
        LBSR       CHCK             COMPARE
        INC        VALOUT,U         CHANGE SIGNAL
        BNE        LOOP             REPEAT TEST
        TFR        U,S
        PULS       CCR,A,B,U,PC     RETURN
* END CALLING ROUTINE
*
*
* SUBROUTINE, PINIT, TO INITIALIZE PIA FOR DAC INTERFACE
*
PINIT   PSHS       A
        CLRA
        STA        PIA2AC           SET UP DDRA
        COMA
        STA        PIA2AD
        LDA        #$04             SET UP DATA
        STA        PIA2AC           REGISTER ADDRESS
        PULS       A,PC
```

```
• END OF SUBROUTINE, PINIT.
•
PIA2AC  EQU     $8052           CONTROL REGISTER
PIA2AD  EQU     $8050           DATA REGISTER
•
•
•
• SUBROUTINE, GINIT, INITIALIZES PIA FOR ADC INTERFACE
•
GINIT   PSHS    A
        CLRA
        STA     PIA1AC          SET UP CONTROL REG A
        STA     PIA1AD          SET UP DATA REG A
        STA     PIA1BC          SET UP CONTROL REC B
        STA     PIA1BD          SET UP DATA REG B
        LDA     #$04
        STA     PIA1AC          SET UP A DATA REG ADDR
        STA     PIA1BC          SET UP B DATA REG ADDR
        PULS    A,PC
• END SUBROUTINE, GINIT.
•
PIA1AC  EQU     $8042           A CONTROL REG ADDR
PIA1BC  RQU     $8043           B CONTROL REG ADDR
PIA1AD  EQU     $8040           A DATA REG ADDR
PIA1BD  EQU     $8041           0 DATA REG ADDR
•
•
•
• SUBROUTINE, DAC, OUTPUTS CURRENT ACCA VALUE VIA PORT A
•
DAC     PSHS    A
        LDA     VALOUT,U
        STA     PIA2AD
        PULS    A,PC
• END OF SUBROUTINE DAC
•
• SUBROUTINE, ADC, CONVERTS ANALOG SIGNAL TO
• DIGITAL EQUIVALENT.  CHECKS ONLY HIGH ORDER
• 8 BITS OF POSSIBLE 12 BIT DIGITIZED VALUE
•
ADC     PSHS    B,A,CCR
        LDA     #$36            LOWER START -
                                CONVERSION LINE
        STA     PIA1AC
        LDB     #$3E            PULL IT HIGH
        STB     PIA1AC          NOW LOW
        STA     PIA1AC
• CLEAR CRA7 TO SET UP END OF CONVERSION
• TEST BY DUMMY READ
        LDA     PIA1AD
• NOW WAIT TILL CONVERSION IS COMPLETE
WAIT    LDA     PIA1AC          CA1 HI ?
        BPL     WAIT            NO
• SET ADC VALUE INTO ACCB AND RETURN
        LDB     PIA1AD          YES
        STB     VALIN,U
        PULS    CCR,B,A,PC
• END OF SUBROUTINE ADC.
•
• SUBROUTINE, CHCK, CHECKS DIGITIZED VALUE FROM ADC
• WITH ORIGINAL VALUE FROM ACCA CONVERTED TO ANALOG
```

```
* /COUNT/ IS SET TO NONZERO IF NO MATCH
*
*
CHCK    PSHS    B,A,CCR
        LDA     VALOUT,U        GET ANALOG VALUE
        CMPA    VALIN,U         COMPARE TO DIGIT-
        BEQ     OUT             IZED VERSION
        INC     COUNT,U
OUT     PULS    CCR,A,B,PC
* END OF SUBROUTINE CHCK.
*
*
*
VALOUT EQU      -1              DAC OUTPUT VALUE
VALIN  EQU      -2              ADC INPUT VALUE
COUNT  EQU      -3              ERROR FLAG
```

4.1.3 串行接口

微处理所用的串行接口通常都是串行数据流的形式，如图 4.9 所示。字符的格式一般都要设置一个起始位，按着为 8 位信息位，最后为 1 或 2 个终止位。图中所示的“传号”(mark）或二进制数值 1 的状态表示到外设和 MPU 的传输处于空闲方式（即没有字符在传送）。需要注意的是起位总要求是一个“空号”（space）或为 0 态。按上述格式传输每个字符时需要有11位数字信号。除去常用的这种格式外，也还有其它的串行数据格式，在此不再叙述。

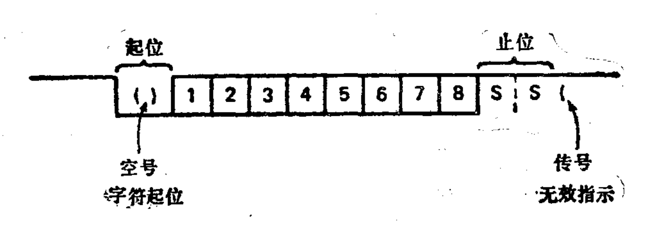


图4.9　异步终端的字符格式

如果要求按图4.9的字符格式输出串行数据流，并按图4.1的单线输出线路配置，其所需软件程序如表4.2所示。该程序输出字符速率是每秒10个字符。使用延迟子程序形成空号间隔脉冲输出为9.1ms（此时，需每秒10字符的接口）。这时6809工作频率为 1 MHz。

异步通信接口器件——MC6850

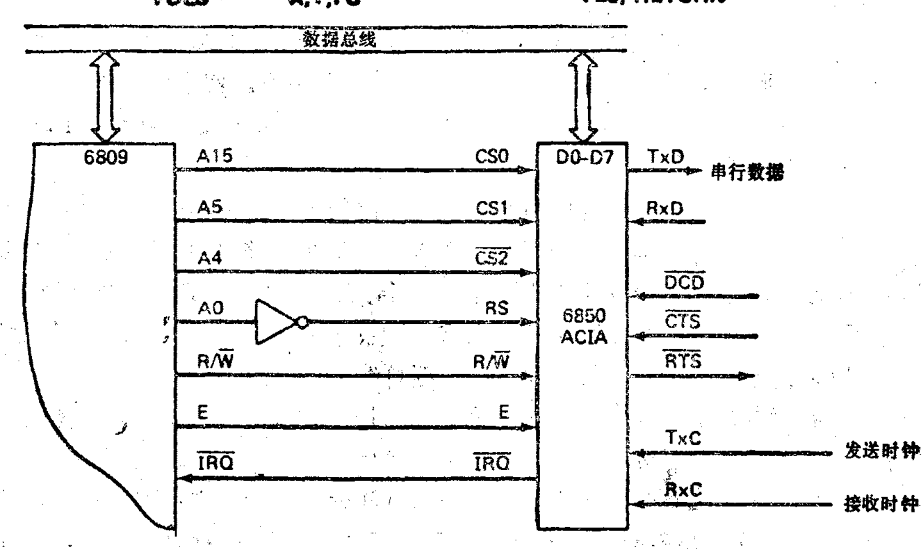
这里简要介绍一下68系列中常用的性能较高的串行数据传输器件，即异步通信接口——MC6850，或称ACIA器件（详细内容见参考资料〔1〕）。该接口器件设有 8 位并行数据总线，可和MPU交换数据，并把这 8 位数据转换为串行方式送给与外部相连的发送数据端 Txd，如图4.10所示。从外部接收数据给MPU时，需经过接收数据端 Rxd 进来，串行数据在ACIA内部被自动地转换为并行数据，并送到 8 位数据总线上。

6850器件中设有几个重要的为交接方式工作的控制信号线，其中有“清除发送”CTS、“请求发送”RTS和“数据载波检测”DCD三信号。外部设备如调制解调器（MODEM）或其它串行数据设备可以使用这些信号，在6850内部设置其状态寄存器的内容，以便给出特定标志通知MPU动作。串行数据设备可以使用不同的发送和接收时钟，因此 6850 可以接受这种不同频率的发送和接收时钟，能够分别取出串行数据的内容。

表4.2 串行数据输出

```
* THIS ROUTINE SENDS A SERIAL CHARACTER OUT THROUGH
* THE LEAST SIGNIFICANT BIT D(0) OF LOCATION $8001.
* USES DATA WHERE ADDRESS IS ON THE STACK AT /CHAR/
* STACK PICTURE ON ENTRY AND EXIT
*        U+0        OLD STACK MARKER
*        U-2        ADDRESS OF CHARACTER
*        U-4        OUTPUT KEY (8001)
*
* SUBROUTINE BODY
OUTTY   PSHS    B,A,CCR
        LDB     #11             SET COUNTER TO 11
        LDA     [CHAR,U]        GET CHAR INTO ACCA
        ANDCC   #$FE            CLEAR CARRY FLAG
        ROLA                    CARRY INTO A(0)
SENDIT  STA     [OUT,U]         TRANSMIT BIT
        LBSR    DELAY           WAIT 9 MILLISECONDS
NEXT    RORA                    POSITION NEXT BIT
        ORCC    #$01            PLACE STOP BITS
        DECB                    COUNTDOWN
        BNE     SENDIT
        PULS    CCR,A,B,PC      RETURN
* END OF SUBROUTINE
* STACK PARAMETER OFFSET VALUES

CHAR    EQU     -2              ADDRESS OF CHARACTER
OUT     EQU     -4              OUTPUT KEY
* DELAY ROUTINE OF 9.1 MILLISECONDS
* ASSUMES NO INTERRUPTS OR WAITS ON MPU
* COMPUTE DELAY WITH CLOCK CYCLE TIMES
*               #$0A TIMES #$64
DELAY   PSHS    Y,X
        LDX     #$0A            INITIALIZE X1 COUNTER
X1      LDY     #$64            INITIALIZE X2 COUNTER
X2      LEAY    ,-Y             COUNTDOWN X2
        BNE     X2              ZERO?
        LEAX    ,-X             COUNTDOWN X1
        BNE     X1              ZERO?
        PULS    X,Y,PC          YES, RETURN
```



8021　R/W̄ = 1 时，为读状态
8020　R/W̄ = 0 时，存入发送数据寄存器

图4.10　6809和6850（ACIA）接口线路

图4.11是用ACIA发送时的程序流程图，其程序如表4.3所示。程序中使用了控制信号，并检查6850状态字中的TDRE状态位，通过它了解发送数据寄存器是否为空位，如果ACIA从MPU接收了8位数据，则该位不空。载波检测标志位$\overline{DCD}$表示正在发送载频。$8021单元表示ACIA的状态，单元$8020内容表示从MPU寄存器ACCB接收的8位并行数据。

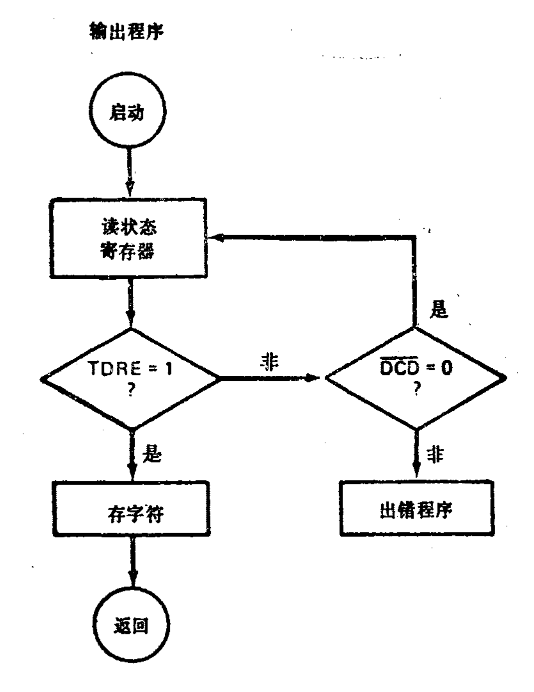


图4.11　ACIA发送程序流程图

表4.3　ACIA发送数据程序

```
* THIS ROUTINE OUTPUTS A DATA WORD THROUGH AN ACIA, CHECKING
* THE CARRIER DETECT FLAG, DCD, BRANCHING TO AN (ERR)
* ERROR ROUTINE IF CARRIER IS LOST.  THE SUBROUTINE USES
* MPU REGISTERS A, B, AND CCR.
*
*
*        ACIA     STATUS REGISTER           8021
*        ACIA     TRANSMIT DATA REGISTER    8020
*
*
* SUBROUTINE BODY
ACIATR   PSHS     B,A,CCR                   SAVE MPU REGISTERS
         LDA      ACIAST                    CHECK TDRE BIT
         ASRA
         ASRA
         BCS      TXDRY                     TRANSMIT DATA
         ASRA                               REGISTER EMPTY?
         ASRA                               NO
         BCC      ACIATR                    CHECK DCD FLAG
         LBRA     ERR                       CARRIER OK?
TXDRY    STB      ACIADA                    NO
         PULS     CCR,A,B,PC                YES, READY TO SEND
                                            RETURN
ACIAST   EQU      $8021
ACIADA   EQU      $8020
```

4.1.4 标准接口

微处理器范围使用的通用标准接口尚有待设计制定。当前的大多数标准或者是实际上被承认的，或者是建议采纳的。作为实际上的标准有S-100总线、MULTI-BUS总线（Intel公司）、VME总线和LSI-11。目前建议采纳的标准有CAMAC IEEE583总线、IEEE488和RS-232标准。除RS-232标准之外，所有这些总线都是并行的I/O标准。

标准接口的的技术指标通常应该包括排线表或引线表、状态图、引线定义和信号参数容限。当前流行的多数标准都具有这些资料，虽然不是所有内容规定得都很明确。例如，S-100总线还有一些没有定义的引线，明显的缺点是地线和电源引线都较少，而且主要适合于8080微处理器系列。不管怎样，S-100总线还是很普及的。在微型计算机领域中，许多外部设备都同S-100总线相兼容。IEEE协会还正在研究S-100总线的标准规范。

标准的类型

标准接口的主要类型有两种。一般说，在MPU和I/O之间或者是主从关系；或者是说听关系。前者，指总线主模块产生驱动命令并驱动地址线。单板计算机就是主模块，由主模块控制其它从属模块或I/O板。因为从属模块不能控制总线，存储器和I/O模块都是典型的从属模块。在一个典型接口可能有多个主模块存在，因此在几个主模块同时请求使用总线时，它们之间则需结合通信协议制定标准技术规范。此时总线时钟要作为定时基准，以便在多主模块请求时解决总线争夺问题。

在各个模块间使用说听关系的一个精心制定的标准接口就是488标准。因为总线基本工作在两个主要方式之一，系统控制器必须建立哪个设备在“说”，哪些（包括多个）设备在“听”。同时，系统控制器还能结束网络的工作过程。在488标准中，可有一个或多个听者能获得总线。

某些考虑

当选择某一总线接口时的另外一种考虑就是电源设施分布的结构，特别是使用单板结构的微型计算机系统，更加显得重要。因为在底板上焊接时，需要了解电源是集中型的还是分散型的。集中型电源容易进行调整，分散型电源需要在底板每块插件板上进行调整。应用中选用集中型还是分散型电源分布，这取决于环境的干扰和实际电源走线分布的情况。电源分布的情况是总线标准中有关电气技术指标要说明的一个方面。在总线标准中，一般也规定了电源电压和电流允许变化程度（容限值）。同时对每个电气信号的容限也作了规定。

莫托罗拉公司对68系列的8位机系统制定了EXORciser86总线标准，对16位机或32位机制定了VMEbus总线标准。前者的具体规定见附录14。

4.1.5 RS-232标准接口

大多数机械式的输入/输出设备同微处理器的时钟速率相比是很慢的，而且，一般情况下，同微型计算机系统的距离也是较远的，因此，它们之间进行连接时往往使用串行数据通信接口。还有一种情况就是在许多应用中要求微型计算机系统或微处理器与外部设备之间的连线数目最少。所以RS-232C就是为满足这些条件制定的普及型标准接口。RS-232接口的电压变化范围最小为3 V，最大为25V；而且终端要设有3～7 kΩ的电阻。使用负逻辑信号，关态为逻辑1，开态为逻辑0。数据字符信号的传输设有发送（TRANSMIT）、接收（RE-

CEIVE）和信号地三条引线。RS-232C标准接口的说明见表4.4。RS-232接口标准可能要被RS-422标准来代替，原因是RS-232的电缆长度较短（常用为50英尺或更短些）。

表 4.4 RS-232C接口标准

线路	引线号	电压（V）	信号名称和符号	功能
AA	1	—	保护地 OR	希望使用中有公共地时，使MODEM或耦合器机壳接到控制器机壳。
AB	7	—	信号地 OV	连到控制器上的所有电路建立公共参考点。
BA	2	±12	发送数据TD	发送控制器来的电报调制信号给 MODEM 或耦合器（TTY来）
BB	3	±5 ～ ±25	接收数据 RD	发送 MODEM或耦合器来的电报调制信号给控制器（到TTY）
CA	4	±12	请求发送 RG	当从线路CD接收（+）信号，该线给MODEM或耦合器传送(+)信号，使发射载波和在CB线路上产生接收的回答(TTY来)
CB	5	±5 ～ ±25	清除发送 CTS 或 发送就绪 RS	发送到控制器的信号（+）“清除发送”（有线路相连，发送载波）。RS灯亮。这是对“请求发送信号RG”通过CA线路送给MODEM或耦合器的响应(有延迟)(到TTY)
CC	6	±5 ～ ±25	数据装置就绪 或 MODEM就绪 IT	发送到控制器的信号（+）“数据装置就绪”。MR信号灯亮（到TTY）
CD	20	±20	数据终端就绪 或 上线路 CL	发送到MODEM或耦合器的信号（+），控制线路交换（TTY来）
CE	22	—	振铃信号	—
CF	8	±5 ～ ±25	接收线路信号检波 CD	当收到载波时，发送到控制器的信号（+）。信号为（—），表示无载波，使信号灯CD点亮并复位控制器（到TTY）
—	10	—	无用	—
—	11	—	无用	—

4.2 用MC6829 MMU作存储器扩充

4.2.1 概述

在1960年，Atlas 就考虑过页面方式的虚拟存储原理，后来被应用到近代的中、大型计算机中。差不多到了八十年代，8 位微处理器MC6809加上存储器管理单元(MMU)MC6829，即可实现页面式虚拟存储方式。在微机中，它不需要考虑存储区的结构，程序可以动态分配，可以生成实际物理地址，这在一般的小型计算机中，也未达到。

6809系统，不但是单个芯片的结构设计，其系统设计思想也是 8 位微机中比较好的。

在说明MC6829的工作原理之前，先说明一下虚拟存储的基本原理。

虚拟存储方式的必要性：

（1）当程序员编写程序，对程序所占用的存储器的容量、地址等硬件情况不完全清楚时，为了防止发生程序不能执行的情况，于是就设了虚拟存储方式。

（2）有时经常需要几种程序并行执行，即使各个程序使用了同一个地址，也不能彼此

破坏对方的程序，这就需要采用虚拟存储技术。

（3）主存储器内装入了许多小型的程序，各程序之间占有的地址不连续，有许多空白区。要对这种有许多空白区的主存储器进行整理（动态分配），使能执行大型的程序。

（4）在程序要执行的时候，由于没有主存储器可以分配，需要把程序内的逻辑地址变换为实际物理地址，这时需要实现虚拟存储方式。

上述这些情况，如果有一个可以从同一逻辑地址自动地配置成不同的物理地址的系统时，立即可以解决。这种情况的逻辑地址，处于处理器的地址空间之内，而实际的物理地址，按照页面分配，有时也可能在处理器的地址空间之外。由此，固定地址的子程序，如果被分配在不同的页面内，则就被安排在不同的物理地址之上。

下面举例说明虚拟存储方式的必要性。

设：程序区有1k字节，从$8000地址开始，数据区有1k字节，从$2000地址开始。设可利用的存储器只有从0号地址到$1000地址的4k字节。

由于所设程序中，数据区和程序区是分开的，即使是位置独立的程序，也不能在从0号地址到$1000地址的存储器中分配该程序和数据。

但是，如果把程序区$8000地址分配到存储器的实际物理地址0地址上，把数据区$2000地址分配到实际物理地址中的$800地址上，则就可以执行这个程序了。

程序写的地址称为逻辑地址，即所谓虚拟地址，它和存储器中的实际物理地址是有区别的。

另外，有多种程序要并行处理时，每个程序（任务）都要设置从虚拟地址变换为实际物理地址所需要的数据。

在虚拟存储方式中，程序的段落只需写明逻辑地址，而执行时的物理地址是由操作系统中管理存储器的空白区域和地址变换的系统程序来决定的。因此，在虚拟存储方式中，即使是写在同一逻辑地址上的程序，也可以分配在不同的实际物理地址上。

应用虚拟存储功能，应该执行的程序，即使不用位置独立方法的技巧，也可以自动地进行地址变换。对某个子程序，如果好几个主程序在配置上都要用到这个子程序，也不必要使程序按再入方式执行。这时，子程序虽然在同一逻辑地址上，但根据页面，可以被安排在不同的实际物理地址上。

采用位置独立型程序技术，即使是同一地址的程序，也可以同时执行。这时，写好了的程序，能安排在存储器位置不同的系统中运行。采用虚拟存储方式，也完全能达到这种要求。

应用存储器管理单元（MMU）的虚拟存储技术，能有效地利用系统资源，对多任务和中断实时处理系统是很合适的。

多任务往往是指编辑、汇编、编译、字处理等多作业任务同时进行。

一般情况下，系统能力是由外部输出装置，如打印机、终端、磁盘等之间的数据传送能力来决定的。而处理器几乎处于休止状态。

按照多任务的思想，处理器执行各种类型的处理是在输入输出装置可以使用的瞬间进行数据传送的。这时，还同时进行后台的处理。这样，系统资源的利用就处于非常高效率的状态。而且，不是设置几台同样的系统，而是多数人同时使用同一系统，就好象本人自己独立使用该系统那样，互不干扰，互不影响。

4.2.2 虚拟存储方式的原理

在虚拟存储方式中，虚拟地址空间（处理器的输出地址空间）和物理地址空间（实际硬件存储器所在的空间）的大小是不同的，有下面两种不同的情况：

第一，象MC68000那种处理器，虽然具有16M字节的虚拟地址空间，但实际上并不是实装16M字节的存储器，实际上可能只装128K字节的存储器，而按虚拟存储方式，则具有实装16M字节存储器的处理能力。

第二，象6809这种处理器，只具有64K字节的虚拟地址空间，而实际上却实装存储器256K字节或1M字节，按照这种方式，实际上能使用的不只是64K字节的地址空间，而是要能够使用全部256K字节的实装存储器系统。

在第一种方式中，超过所实装的物理地址空间的存储器直接保留在所用的磁盘等外部存储器中。当把数据从磁盘等外部存储器上移入实装物理地址的存储器中时，需要更改偏移值。外部存储器如果是16M字节，即使实装存储器不是16M字节，应用时有 16M 字节的程序也能执行。

第二种方式又称页面方式。参考图4.12，高速存储器（图中为地址分配RAM）的地址输入线接到虚拟地址线（处理器的地址总线，图中为 A11～A15 ）上，而高速存储器的输出数据，决定了物理地址空间。以 MC6829 为例，虚拟地址的高 5 位连接到高速存储器的地址线上，如果高速存储器输出10位数据，则物理地址空间一下子扩展32倍（$2^{10}/2^{5}=32$）。

在这种高速存储器开始执行之前，需要预先写入数据。这种数据即为地址变换信息，又称为变换表。在该表中，作为地址变换的信息，将包含有各个虚拟地址应该变换为哪个物理地址的数据。

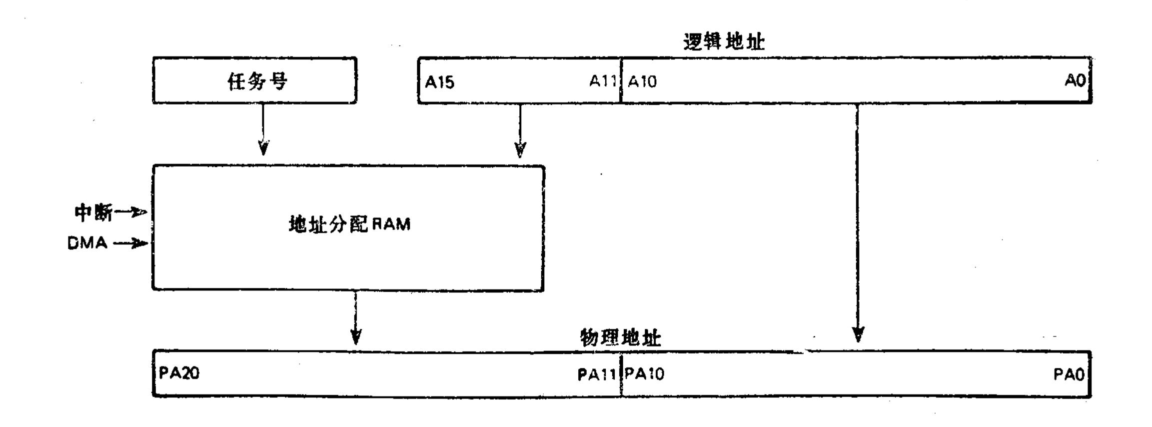


图4.12　从虚拟地址往物理地址的变换

物理地址不一定要把虚拟地址的所有位都进行变换，通常只使虚拟地址的高位数字进行变换，而未被变换的低位，如A_0～A_{10}，仍然可以作为物理地址的PA_0～PA_{10}。

一页的大小，不由高速存储器的地址线决定，而由物理地址线的地址位数构成，不包括高速存储器的地址线位数，在本例中，一页的大小即为A_0～A_{10}形成的2048字节。

4.2.3 内部寄存器的组成及其任务

MC6829（MMU）在每个处理器周期中，根据执行的任务号和处理器形成的虚拟（Virtual）地址生成物理（Physical）地址。参见图4.13。

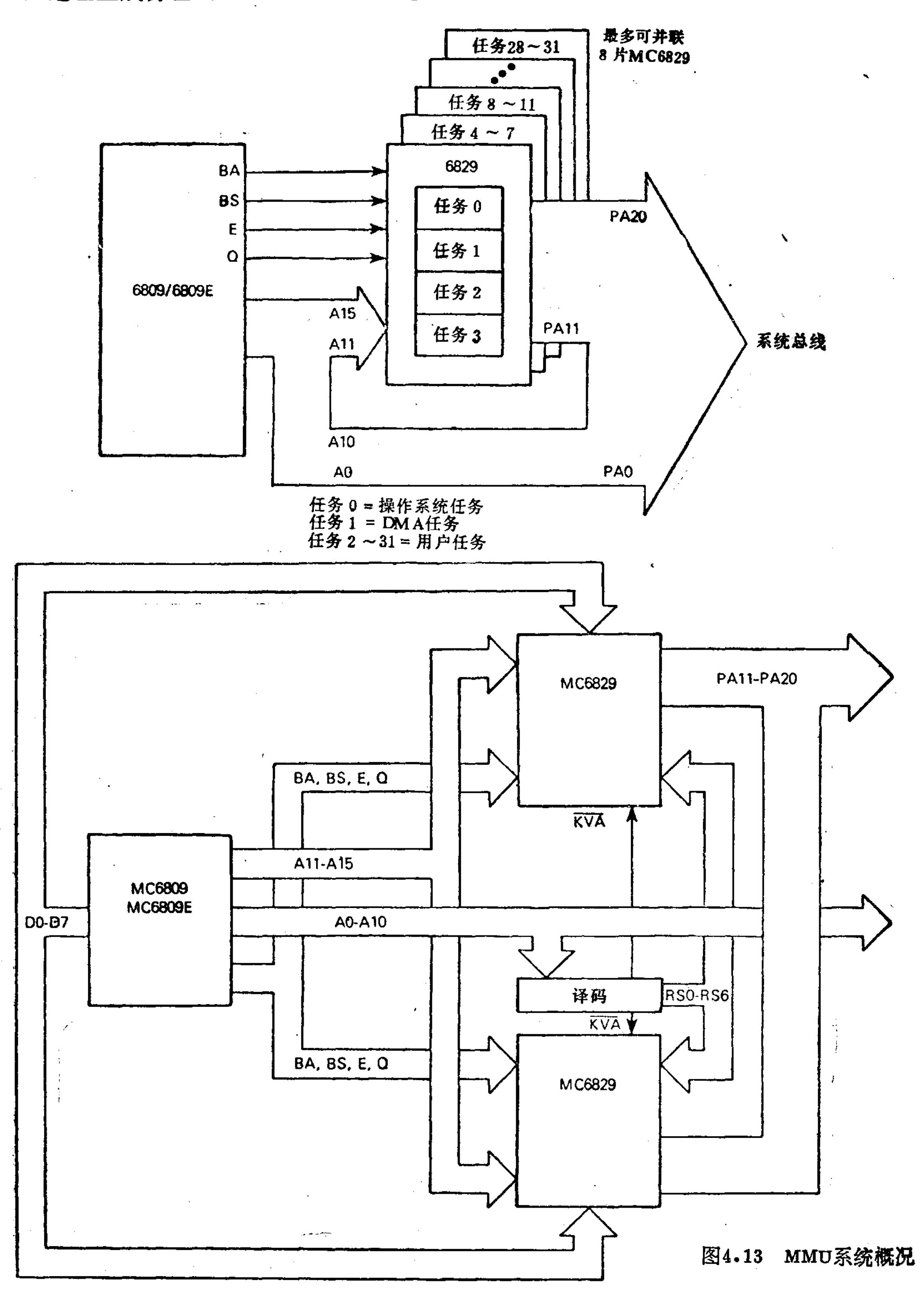


图4.13 MMU系统概况

在说明实际工作原理之前，先对形成这种物理地址所需要的MC6829内的控制存寄器进行说明。参见图4.14和图4.15。

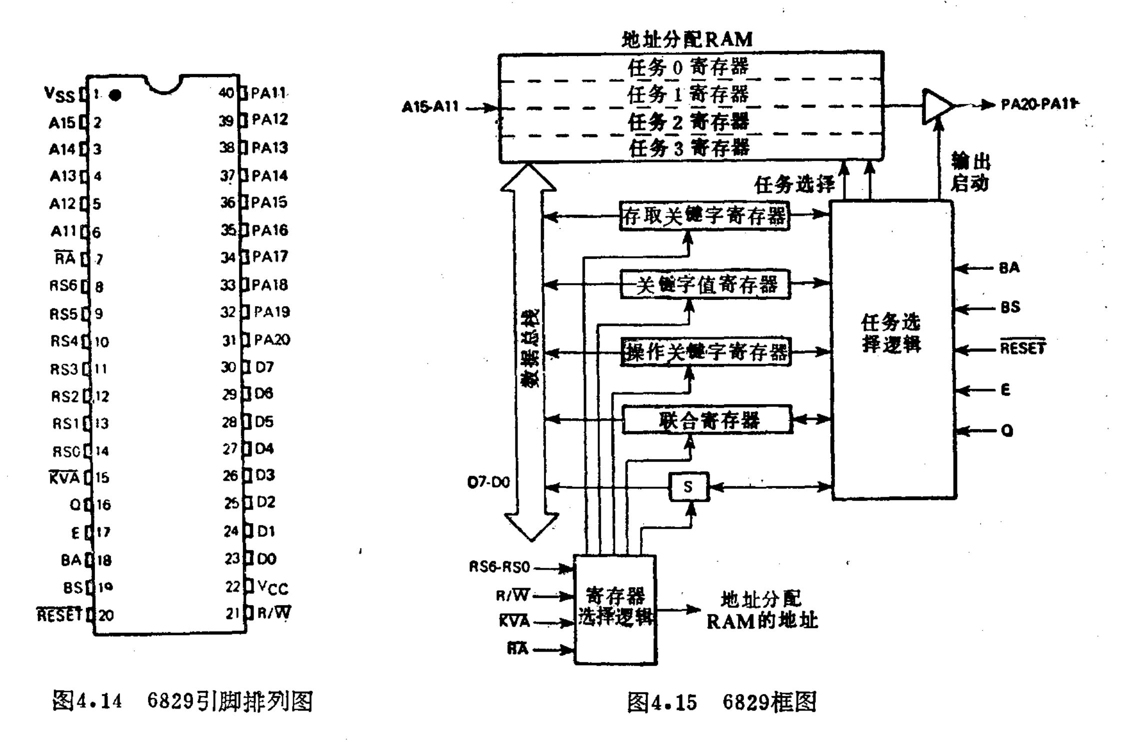


图4.14　6829引脚排列图　　　　图4.15　6829框图

（1）变换表格管理存储器（$ 0～$ 3F）

（虚拟地址变换为物理地址所需变换数据的存储器）

每个MC6829有4个变换表，其变换表格管理存储器（Access Key “Window” 存取关键字窗口）使用寄存器选择（RS）的$0～$ 3F地址的64个字节的地址。设每个字有16位，而高6位不用，读取时为零，则一个变换表有32个字，此32个字的变换表信息只有在系统标志（$48）的“S”被置位，虚拟地址A11～A15全部为1时，才能改写。6809系统最多可并联8个MC6829（MMU），所以有32个变换表，而且所有MMU的变换表全部安排在相同的地址$ 0～$ 3F上，全由$ 0～$ 3F来进行读写。因此，需要指定哪个MMU芯片的哪个变换表需要进行改写，这由存取关键字来指定，后面将讲到。在操作系统程序中，当存取关键字设定之后，被改写的变换表也随之决定。这时，首先在此变换表的地址0、地址1上写入虚拟地址$0000～$07FF变换为物理地址中几号地址的信息，例如，假设写的是$ 1F0，那么，对于$0000～$07FF范围的虚拟地址对应的物理地址则为$ 1F0000～ $1F07FF范围内的2K字节。同样，在变换表的地址2、地址3上，写入了虚拟地址$0800～$0FFF变换为物理地址的变换数据。对64K字节的虚拟地址全部要变换为物理地址时所需要的变换数据为32个字（即64个字节）。所以，如果有一个任务，需要64K字节的存储器容量时，则需要在这32个字中全部写上变换数据。

在一个MMU芯片中，设有4个为任务所用的变换表，总共有128个字的变换数据，通过$ 0～$ 3F这64个地址窗口进行写入。在有8个MMU的系统中，有32个变换表，则有1024个字（2048个字节）的变换数据，皆通过$ 0～$ 3F地址窗口写入。如果一个任务占用一

个变换表，接有8个MMU的系统，则可带32个任务，对应所用的任务的变换表，全要写入各个MMU中。

（2）关键字值寄存器（$40～$47）

（指定选中哪个MMU的寄存器）

在8个MMU组成的系统中，为了决定选择哪个MMU，设置了关键字值寄存器。每个MMU中皆有一个关键字值寄存器，它是一个三位的寄存器。当系统中只有一个MMU时，关键字值寄存器预置为0即可，在有8个MMU的系统中，各个MMU可以预置为$0～$7。

系键字值寄存器在$\overline{KVA}$为“0”、RS0～RS6为$40～$47时，即可选中。在各个MMU中，关键字值寄存器置位关键字的号码时，可以预置为$0～$7数值，但绝不能在两个以上的MMU中写入同一个关键字值。

寄存器的选择见图4.16。MMU中寄存器模块结构见图4.17。

（3）联合寄存器（$49）

（任务转移所需机器周期时间的寄存器）

从操作系统（任务0）转移到其它任务时，当执行0号任务的最后指令JMP或RTI完成时，必须同时转移到新的任务。因此，在执行0号任务时，就把某个机器周期时间，在适当的程序指令地址处，写入到联合寄存器中。一般在转移指令的紧前面，编写这几条写入指令。写在联合寄存器中的数据，每过一个机器周期，就自动减1（所以联合寄存器是一个自动减1计数器），到全部数据减为“0”时，就转移到新的任务上。

$\overline{RA}$	$R/\overline{W}$	$\overline{KVA}$	RS6	RS5	RS4	RS3	RS2	RS1	RS0	寻址的寄存器
1	×	×	×	×	×	×	×	×	×	未用
0	×	1	1	0	0	0	×	×	×	未用
0	1	0	1	0	0	0	×	×	×	读关键字值寄存器
0	0	0	1	0	0	0	×	×	×	写关键字值寄存器
0	×	×	0	n	n	n	n	n	n	MMURAM的字节nnnnnn（注1）
0	0	×	1	0	0	1	0	0	0	未用（注2）
0	0	×	1	0	0	1	0	0	1	写联合寄存器
0	0	×	1	0	0	1	0	1	0	写存取关键字
0	0	×	1	0	0	1	0	1	1	写操作关键字
0	1	×	1	0	0	1	0	0	0	读S位（注3）
0	1	×	1	0	0	1	0	0	1	读联合寄存器（注3）
0	1	×	1	0	0	1	0	1	0	读存取关键字（注3）
0	1	×	1	0	0	1	0	1	1	读操作关键字（注3）
0	×	×	1	0	0	1	1	×	×	未用
0	×	×	1	0	1	×	×	×	×	未用
0	×	×	1	1	×	×	×	×	×	未用

注 1. 只有在关键字值寄存器等于存取关键字寄存器头3位内容时，MMU的RAM才可进行存取。存取关键字寄存器的低2位内容决定要进行存取的任务（$R/\overline{W}$）。

2. 只读8位。

3. 只有在关键字值寄存器内容等于存取关键字寄存器头3位时，S位、联合寄存器、存取或操作寄存器才可以读出。这样就保证只有一个MMU对读出这些单元内容的请求产生响应。

图4.16 寄存器的选择

寄存器	D7	D6	D5	D4	D3	D2	D1	D0	逻辑地址
00							PA20	PA19	\$0000-\$07FF
01	PA18	PA17	PA16	PA15	PA14	PA13	PA12	PA11	
02							PA20	PS19	\$0800-\$0FFF
03	PA18	PA17	PA16	PA15	PA14	PA13	PA12	PA11	
04							PA20	PA19	\$1000-\$17FF
05	PA18	PA17	PA16	PA15	PA14	PA13	PA12	PA11	
⋮									
3E							PA20	PA19	\$F800-\$FFFF
3F	PA18	PA17	PA16	PA15	PA14	PA13	PA12	PA11	
40						KV MMU0			每个MMU只有一个关键字值寄存器且所有关键字值寄存器都位于这个区域
41						KV MMU1			
42						KV MMU2			
43						KV MMU3			
44						KV MMU4			
45						KV MMU5			
46						KV MMU6			
47						KV MMU7			
48								S	系统/用户标志位
49						Fuse			地址分配开关保险
4A					存取关键字				通过寄存器＄0～＄3F立即存取的任务
4B					操作关键字				当前任务
4C									未定义
⋮									
7F									

（寄存器00～3F：存取关键字窗口）

注意：
1. ＄4C～＄7F字节的内容都未定义，对任何读/写都不响应。
2. 在复位时，存取、操作关键字和关键字值寄存器清0，S位置位。
3. 所有寄存器中未用位读出总为0。
4. 只有当$\overline{KVA}$= 0 时，＄40～＄47单元才能存取。
5. 在配置多片MMU的系统中，凡是关键字值寄存器和存储关键字头3位符合的MMU，将响应处理器读＄48～＄4B的内容，当处理器向这些寄存器写入时，将会同时使数据写到MMU寄存器中。

图4.17　MMU寄存器模块

在联合寄存器中，写入多少机器周期为好？下面用实例说明：

```
LDA  #4      把应该写入联合寄存器中的机器周期数取到累加器A中
STA  FUSE    把周期数写到联合寄存器中
JMP  USER    在JMP指令执行结束的同时转移到用户任务
```

	操作系统					任务n
处理器的动作	STA FUSE 的执行	取出JMP指令	取出JMP指令的目的地址的高位字节	取出JMP指令的目的地址的低位字节	$\overline{VMA}$	任务n的操作码
联合寄存器的内容	0	4	3	2	1	0

n是操作关键字的内容

（4）存取关键字寄存器（$ 4A）

（指定哪个MMU中的哪号任务的寄存器）

在改写最多有32个任务的变换表时，用存取关键字寄存器选择其中的一个变换表使用。

存取关键字寄存器由5位构成。高3位和各个MMU的关键字值寄存器的内容自动地进行比较，只有相一致的MMU芯片才被选中，低2位是从所选中的MMU内的4个任务中选择1个任务。例如：MMU 0的关键字值寄存器的内容是$ 0，如果存取关键字寄存器的内容为$ 3，那么MMU 0的第4号任务被选中。MMU 0的第4号任务用的虚拟地址变换为物理地址的变换信息出现在$ 0～$ 3F的MMU寄存器区域。这样，就可开始更改$ 3号任务的变换表。

（5）操作关键字寄存器（$ 4B）

（任务号寄存器）

操作关键字寄存器是存储应该执行的任务号码的寄存器，由5位数据构成。这5位数据和存取关键字寄存器一样，是由最多8个MMU组成的32个任务之中选择其中一个任务号的内容。操作关键字寄存器的内容，即使因中断向#0号任务恢复，或是向#1号任务（DMA方式）转移，都不会发生变化。

（6）未使用的寄存区（$ 4C～7F）

未使用的寄存器区的所有位数，都未定义。

4.2.4 实际处理情况

（1）系统的启动

参看图4.18。MC6829的硬件设计为：当总清信号$\overline{RESET}=0$时，系统进行初始化，这时选中#0号任务的最终页面，而与#0号任务的变换表无关，物理地址固定选择为最高位页面。也即，在$\overline{RESET}$后、处理器产生$ F800～$ FFFF的虚拟地址，而物理页面为$ 3FF(因为PA_{11}～PA_{20} 10位全部为1)，所以产生的物理地址为$ 1FF800～$ 1FFFFF（$ 3FF × 2^{11} = $ 3FF × $ 800 = $ 1FF800）。因此，处理器的$\overline{RESET}$的中断向量地址为物理地址上的$ 1FFFFE～$ 1FFFFF地址，初始化程序必须安排在物理地址$ 1FF800～$ 1FFFFF上。同时，因为$\overline{RESET}$的中断向量在物理地址$ 1FFFFE～$ 1FFFFF中，此地址中的内容即表示出虚拟地址$ F800～$ FFFF的地址范围在哪里。接通电源执行$\overline{RESET}$时，MMU内的系统标志位“S”自动地置位，排在物理地址最终页面的操作系统程序，可以把数据写入所有的MMU内的寄存器中。这样一来，当RESTART（再启动）时，在存于物理地址的最高位页面上的操作系统程序就可以执行。

（2）MMU物理地址的安排

必须把MMU中所有的寄存器的地址安排在物理地址的最高位页面上。MMU中的寄存器本身不进行地址分配，必须用选择寄存器存取信号$\overline{RA}$把MMU中的寄存器安排在最终的页面之内。各个MMU需要有128字节的存储器空间。因为采用了关键字值寄存器和存取关键字寄存器，虽然在实际系统中配置有8个MMU芯片，同样也可以只用128字节的存储空间来解决。参见图4.19。

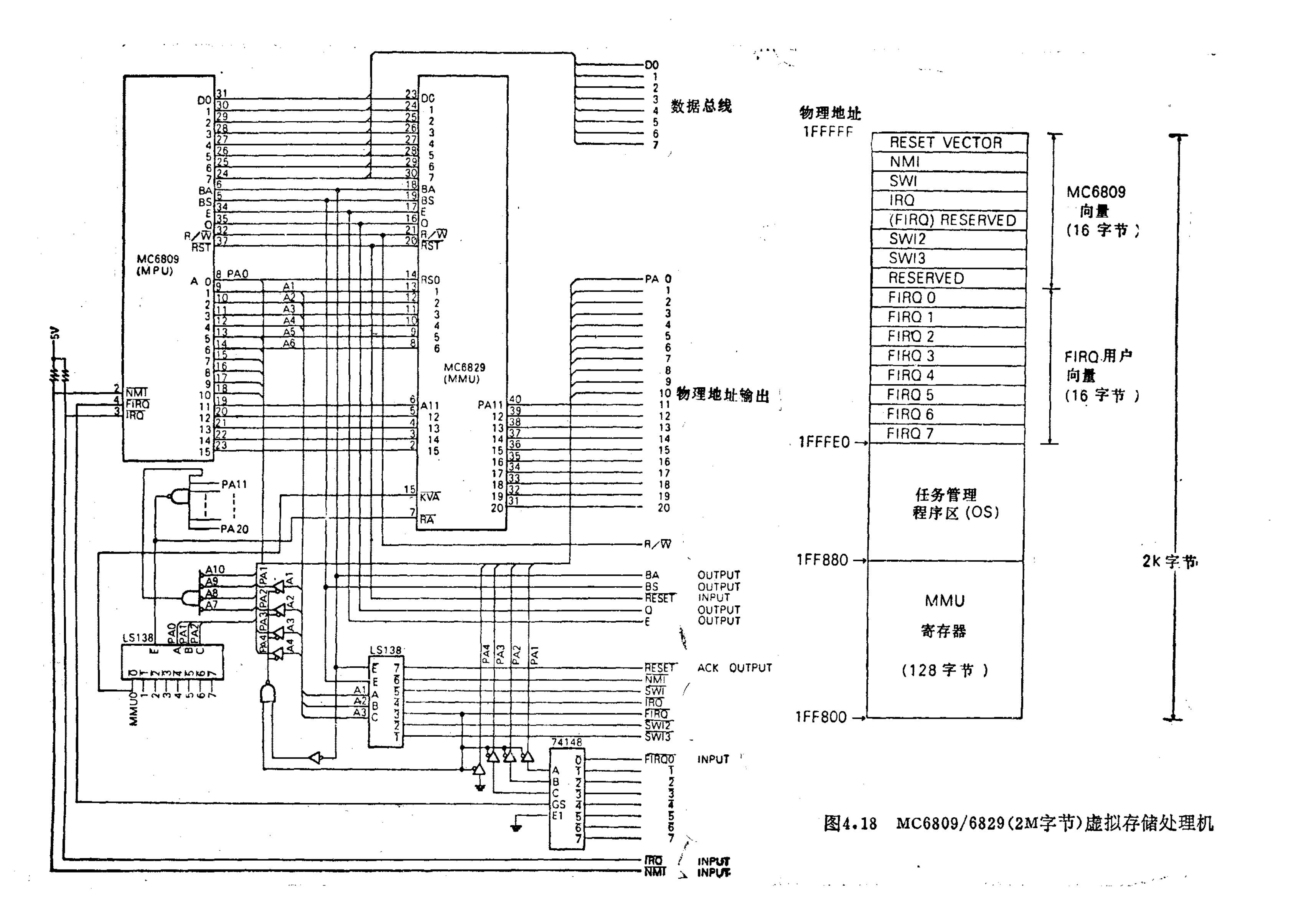


图4.18 MC6809/6829(2M字节)虚拟存储处理机

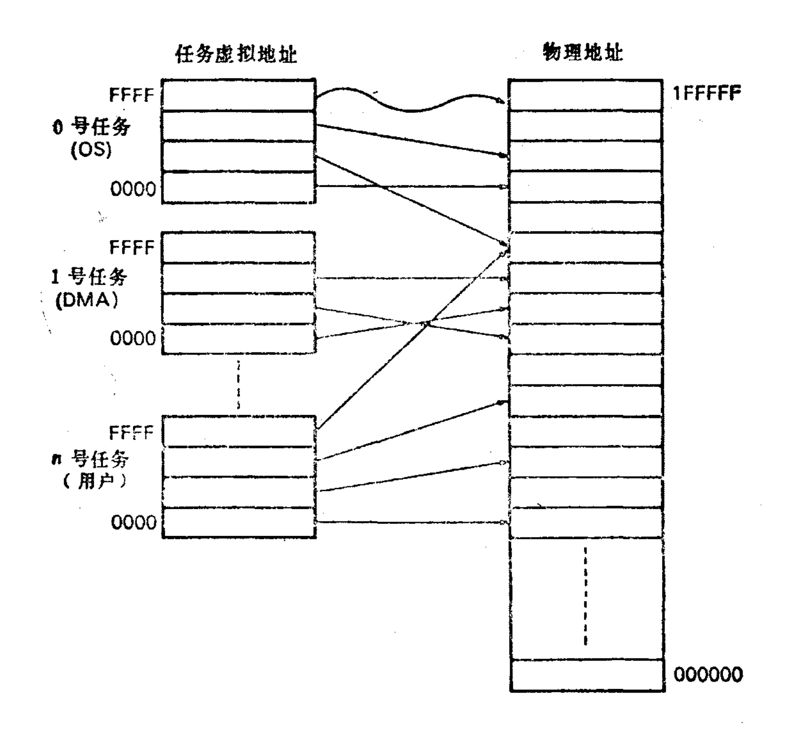


图4.19　物理地址生成举例

$\overline{RA}$信号是用物理地址PA_{11}～PA_{20}相“与”及最高位页面上存放操作系统和程序ROM的位置来设计生成的。一般情况下，把2K字节存储空间的低位作MMU的寄存器用，而把高位作为程序ROM使用，这种方法较易实现。

在图4.18线路中，MMU的物理地址为$1FF800～$1FF87F，该MMU的寄存器块在系统标志“S”被置为1的#0号任务中，处理器生成的虚拟地址（A_{11}～A_{15}）全部为“1”、且$\overline{RA}$为有效时，即可被读写。

同时，RS0～RS6是MMU的寄存器的选择信号端，一般接在处理器的地址线A0～A6之上。按照这样接的全部MMU内的寄存器，只有在#0号任务出现时，才能存取，而在其它任务中，按照存储器存取的选择原则，是不可能对其内容进行改写的。

（3）软件中断的任务

在MC6809/MC6829系统中，如果要从#0号任务的操能系统转移到用户任务时，如果系统标志“S”已被清零，即使存取物理地址为$1FF800～$1FFFFF，也不能改写MMU内的寄存器内容；同时，从软件上讲，为了从用户任务恢复到操作系统（#0号任务），只有执行软件中断SWI，SWI2或SWI3才能实现。因此，要根据操作系统的程序，对用户程序所用的物理地址空间（包括用户任务所用的最高位页面$3FF），使用MMU内所设的硬件，禁止用户任务本身来改变MMU内寄存器的内容。

在使用MC6809/MC6829做的虚拟存储的处理机中，当用户程序要调用系统程序内的部分输入输出子程序时，也可以利用软件中断进行。$\overline{RESET}$总清之后，任务号的更新流程图如图4.20所示。

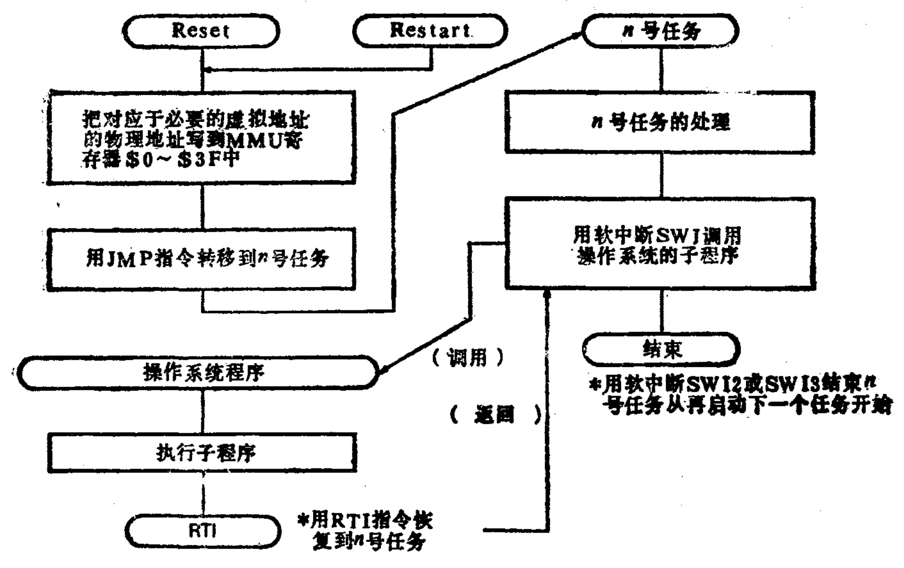


图4.20　任务号更新流程图

MC6809中，有三种软件中断。在用虚拟存储的处理机中，当由用户任务过渡到操作系统（# 0号任务）时，只能用软件中断指令。另外，在只有一种软件中断的情况下，可以做到调用操作系统程序中的子程序，或者结束执行中的用户任务。但是，在每次做时，必须检查转移状态时的寄存器中的内容；而且还需要做“在执行结束完全没错”或者“产生错误”的检查。但是，如果使用三个软件中断，就不必要做这种检查。下面给出软件中断的分类：

SWI　调用操作系统内的子程序

SWI 2　无错，结束任务

SWI 3　有错，退出（Abort）用户任务

（4）往用户任务进行切换

参看图4.21。根据操作系统为用户程序配置所需的存储器，如果向MMU内的$ 0～$ 3F中的虚拟地址和物理地址变换用的地址变换RAM（管理用寄存器）写完之后，就要向联合寄存器中写入在任务切换时的机器周期数。联合寄存器使用每过一个机器周期减 1 的计数器。当计数器的内容为 0 时，则执行任务的切换。这时的状态要对地址变换RAM加以说明。

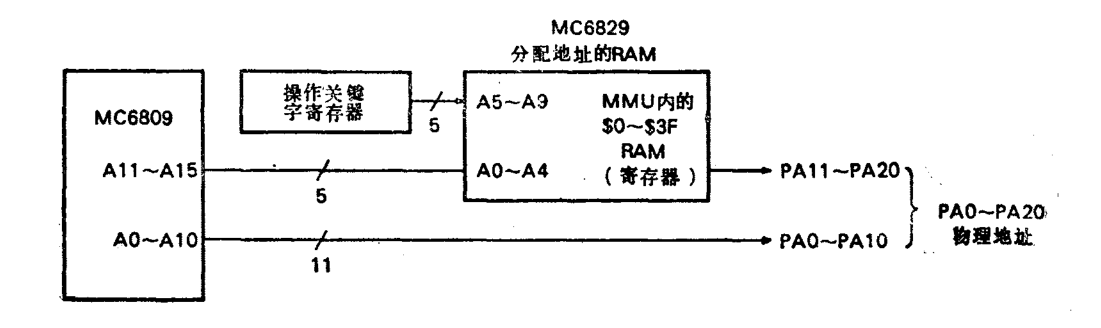


图4.21　物理地址生成中的结构

在操作系统中，变换表中写有地址变换RAM中的数据状态，在地址变换RAM的A5～A9线上传送存取关键字的内容，而物理地址PA11～PA20的各位保持为1状态。同时，地址变换RAM的A0～A4线接在寄存器选择端RS0～RS4上，数据总线接在地址变换RAM中的数据输入端，并进行写入。在地址变换RAM中写过数据之后，就可以向联合寄存器中写入任意值（在用扩充寻址方式的JMP指令时为4），如果联合寄存器内容递减计数到0，则地址变换RAM上的A0～A4就被切换到处理器的A11～A15端，而地址变换RAM的A5～A9就被切换到操作关键字寄存器的输出端，则地址变换RAM的输出PA11～PA20即启动工作。若A11～A15全部为0，则地址变换RAM的输出端PA11～PA20上将出现用操作系统程序写入在MMU寄存器地址$ 0，1中的数据。同样，如果A11～A15全部为1，则MMU寄存器地址$ 3E，3F中的内容将出现在PA11～PA20上。

按上述方式生成的物理地址的内容，和以往的地址总线一样，可以发到存储器和外围芯片等各种输入输出装置中去。这种情况的代表者即为上面（2）中讲到n号任务。

如果对应于生成的物理地址的存储器或外围芯片都不存在时，则处理器就失去了应该读出/写入的对象，这种状态可以作为“总线错误状态发生”而检测出来。但在MC6809中，没有受理总线错误状态的设计。如果被丢失的是堆栈区域，那么，在用户程序中的子程序调用，由于寄存器内容压入堆栈区的保留工作不能进行，结果使用户程序不能继续执行下去。因此，在以往的单一任务系统中，可由操作员再发出$\overline{RESET}$总清信号，而在虚拟存储多任务情况下，执行$\overline{RESET}$不能全部恢复所有的任务，只有退出有问题的任务才行。全部任务退出当然是不合理的。因此，如果发生“总线错误”，在处理器中要产生硬件中断，把正在执行中的用户程序退出去。

为把执行退出的情况报告给用户，需要采用实时监视总线的方法。这种方法就是：如果发生了“总线错误”，先把总线的状态取入先进先出（FIFO）寄存器，再用操作系统来分析先进先出寄存器的内容。

（5）往DMA处理任务的切换

为了实现更高速的多任务系统，在MC6829中，把DMA处理固定作为#1号任务进行特殊处理。一般，在多任务系统中，和磁盘、通道等输入/输出设备之间传送数据时，需要等待时间，这时，可利用DMA方式，把要传送的数据放在DMA控制器中，一有可能，即把这些数据进行DMA传送。在DMA传送期间，处理器可执行其他任务。这样，就使处理器不因为等待外设传送数据而耽误很多时间，使高速多任务执行得以实现。这时，需要用户任务和DMA处理任务之间的切换不要借助于操作系统能自动切换。

在MC6809/MC6829的系统中，因为MC6809/MC6809E对于$\overline{DMA}/\overline{BREQ}$或$\overline{HALT}$输入所作的回答为：输出BS＝1，BA＝1。所以，如果用MC6829监视各个机器周期，检查是否BS＝1，BA＝1，就可不借助于操作系统，自动地高效率地进行任务的切换。MC6809/MC6809E和MC6844DMA控制器相连结时，MC6844最多可处理4通道的DMA，MC6809/MC6809E和MC6829连接时，只能处理一通道DMA，但无论怎样，比借助于操作系统的数据传送的速率要快得多。

MC6809/MC6829/MC6844的接线图如图4.22。使用MC6844（DMAC）时，DMA-GRNT信号可以用BS和BA相“与”的门电路产生。用MC6829/MC6844进行DMA传送时，不管是用$\overline{HALT}$还是用$\overline{DMA}/\overline{BREQ}$哪一个输入，都可以用这个电路进行处理。

图4.22 MC6809/MC6829/MC6844基本接线图（省去寄存器选择、选片电路未画出）

（6）禁止往存储器的写入

在物理地址中装入任何种类的用户程序时，都有可能破坏在调试之中的程序或者固定的数据，为了防止这种可能性的发生，可在存储器的插件上面设置RAM/ROM开关，装入程序时，把开关放在RAM位置，装完以后，再把开关放在ROM端，这样，就可使存储器的内容受到保护。若物理地址范围为1M字节，则PA20物理地址还未用上，就可用它作为存储器的保护线送给存储器插件，见图4.23。另外，在处理器中使R/$\overline{W}$信号与PA20相“或”，可不变动存储器插件，用软件的方法来实现RAM/ROM开关。

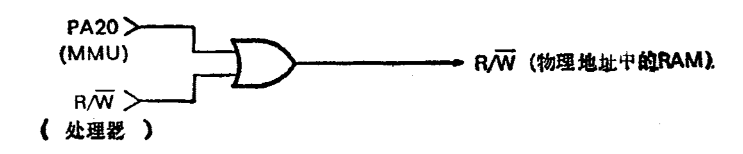


图4.23

（7）程序例

〔公用EQU数据〕

从汇编语句MMU EQU ＄F800起始的一连串EQU数据，是程序例1到例6中所公用的MC6809汇编语言中的EQU指令语句，在实际程序中，需要根据硬件的配置来修正MMU EQU ＄F800的虚拟地址。

```
MMU      EQU   $F800      START OF MMU REGISTERS (IN TASK 0)
MMUO     EQU   MMU+$40    FIRST MMU'S KEY VALUE REGISTER
MMU7     EQU   MMU+$47    LAST MMU'S KEY VALUE REGISTER
SBIT     EQU   MMU+$48    SYSTEM/USER FLAG BIT
FUSE     EQU   MMU+$49    MAP SWITCH COUNT-DOWN REGISTER
ACCESS   EQU   MMU+$4A    ACCESS KEY
OPERAT   EQU   MMU+$4B    OPERATE KEY
NTASK    EQU   32         NUMBER OF TASKS IN SYSTEM
NPAGE    EQU   32         NUMBER OF PAGES PER TASK
MAXPGE   EQU   $400       MAXIMUM NUMBER OF PAGES IN SYSTEM
PSIZE    EQU   2048       NUMBER OF BYTES IN A PAGE
```

〔程序例 1 〕——MMU0以外的关键字值寄存器的初始给定

从MMU 0 到MMU 7 的 8 个MMU的关键字值寄存器的内容由$\overline{\mathrm{RFSET}}$（总清）来初始给定。但并不是照MMU 1 的关键字值寄存器内容为＄ 1 ，MMU 7 的关键字值寄存器的内容为＄ 7 那样简单重复，而是照程序设定把 8 个MMU的关键字值寄存器的内容初始给定的。

在MMU 0 中，$\overline{\mathrm{RESET}}$后，因为还什么都没有写入，所以其物理地址输出是最高位页面（＄ 3 FF）的＄ 1 FF300～＄ 1 FFFFF。在$\overline{\mathrm{RESET}}$总清之后，所有 8 个MMU，都通过PA 11～PA20输出物理页面＄ 3 FF，但由于各个MMU中的关键字值寄存器的内容被置为不同的数值，所以不能 8 个全部同时进行输出。

在本程序中，MMU 0 继续输出＄ 3 FF的物理页面，驱动物理地址总线。

```
•
•               RESET ENTRY POINT FOR MMU SYSTEM
•
         LDX    #MMU7+1     POINT TO LAST MMU KEY VALUE REGISTER +1
         LDA    #7          INITIALIZE VALUE
KVINIT   STA    ,-X
         DECA
         BNE    KVINIT
•
•               CONTINUE INITIALIZATION
•
```

〔程序例 2 〕——＃ 0 号任务的页面给定

在使用此号任务时，为了把＃ 0 号任务所用的虚拟地址＄0000～＄FFFF的64K字节变换为物理地址的＄ 1 F0000～＄ 1 FFFFF，则要采用初始给定MMU 0 中＃ 0 号任务所使用的变换表（MMU寄存器＄ 0 ～＄ 3 F）程序。当程序中一执行CLR MMU 0 ，以前固定输出 3 FF物理页面的MMU，就开始产生对应于输入的虚拟地址的物理地址。在这种状态下，因为系统标志位“S”还是照样被置位，所以，可以按照虚拟地址＄F800～＄FFFF内的程序，对MMU寄存器进行改写。

```
•
•                   FROM KEY VALUE INITIALIZATION
•
•        NOW INITIALIZE IDENTITY MAP FOR TASK 0
•
         CLR    ACCESS     TALK TO TASK 0 (ALREADY ZERO ANYWAY)
         LDX    #MMU
         LDD    #$3E0      LAST PAGE -32
MOINIT   STD    ,X++
         INCB              QUIT WHEN D=$200
         BNE    MOINIT
         CLR    MMU0       LET MMU #0 GO
         JMP    EXBUG      TRANSFER TO MONITOR (EXBUG09)
```

〔程序例 3〕——把 # 9 号任务的物理页面设定为 # 88

这是把物理地址线PA20作为禁止写入线，把系统中的 # 9 号任务的物理页面分配为 # 88 的程序。并指定该页面的虚拟地址为 $ 1 000～$ 17FF。

```
PROTEC    EQU    $200           write protect bit position (PA20)
          ⋮
          LDA    #9             select task #9 for
          STA    ACCESS         modification
          LDX    #88+PROTEC     write physical page into
          STX    MMU+4          the appropriate register
          ⋮
```

〔程序例 4〕——读出特定任务中 1 个字节数据的程序

这是从任意任务中读取 1 字节数据的子程序。调用该子程序时，将累加器（A）作为对象的任务号，把由累加器（A）指定的任务中所要读取的虚拟地址预先放在变址寄存器（X）中。但是，操作系统的第三页面，若能使该程序执行时，则要令操作系统第三页面成为“自由”状态。从子程序返回时，读到累加器（A）中的数据要存起来，该程序要调用程序例 6 的子程序。

```
FPAGE     EQU    $1000          DEDICATED FREE PAGE
FREE      EQU    4              OFFSET INTO MMU OF FPAGE
•
•         FUBYTE – FETCH USER BYTE
•
FUBYTE    LBSR   GETPAGE        POINT TO PAGE
          LDA    ,X             PICKUP BYTE
          RTS
```

〔程序例 5〕——向特定的任务写入 1 字节的程序

和程序例 4 相反，程序 5 是把存在累加器（B）中的一字节数据，按累加器（A）所指定的任务写入变址寄存器（X）所指定的虚拟地址里去的子程序。

和程序例 4 一样，需要操作系统的第三页面空闲起来。

```
•
•         SUBYTE – SET USER BYTE
•
SUBYTE    LBSR   GETPAGE PLACE USER PAGE IN PAGE 3
          STB    ,X
          RTS
```

〔程序例 6〕——从任务号和存储器地址返回指示器的子程序

这是把由累加器（A）所指定的任务监查用寄存器（$ 0～$ 3 F）的数据块值，用变址寄存器按存储的虚拟地址值计算出来，利用操作系统内的成为自由空闲的一个虚拟页面，即可进行读写的子程序。

```
GETPAGE    PSHS    D, Y            SAVE SOME REGISTERS
           STA     ACCESS          SETUP WINDOW TO TASK
           TFR     X, D            MOVE POINTER INTO ACCUMULATOR
           ASRA                    FIND PHYSICAL PAGE #
           ASRA
           ANDA    #%00111110      MASK ALL BUT PAGE #
           LDY     #MMU
           LDY     A, Y            PICKUP PAGE
           CLR     ACCESS          NOW TALK TO OS MAP
           STY     MMU+FREE        'FREE' OS PAGE
           TFR     X, D            NOW POINT TO OFFSET
           ANDA    #%111           MASK HIGH BITS OF ADDRESS
           LDX     #FPAGE          POINT TO PAGE START
           LEAX    D, X            ADD OFFSET
           PULS    D, Y, PC        RESTORE AND RETURN
```

上述存取其他任务的存储器的方法，是访问存储器的字节数比较少的情况下使用的方法。在读写大量的存储器时，要在改变MMU中寄存器的数值之前，传送2K字节（1页面）时，需要编出更通用的子程序为好。

4.3 6809的系统

4.3.1 6809最小系统

微处理器可以用最少种类的器件组成系统，这反映了该处理器所具有的能力。6809的最小微型计算机系统除自己本身之外是由读/写存储器（RAM）、只读存储器（ROM）和并行接口器件组成。该系统如图4.24所示。系统中所用的6800系列的器件保证都和6809系统兼容。使用的6810可作为128字节的高速读/写存储器，在许多应用中都需要使用这种存储器。系统中使用可编程的1K字节掩膜ROM6830，其功能设有中断向量、中断服务程序以及特殊应用所需的子程序。系统采用6821并行接口（PIA）。整个系统的存储器地址分配情况如

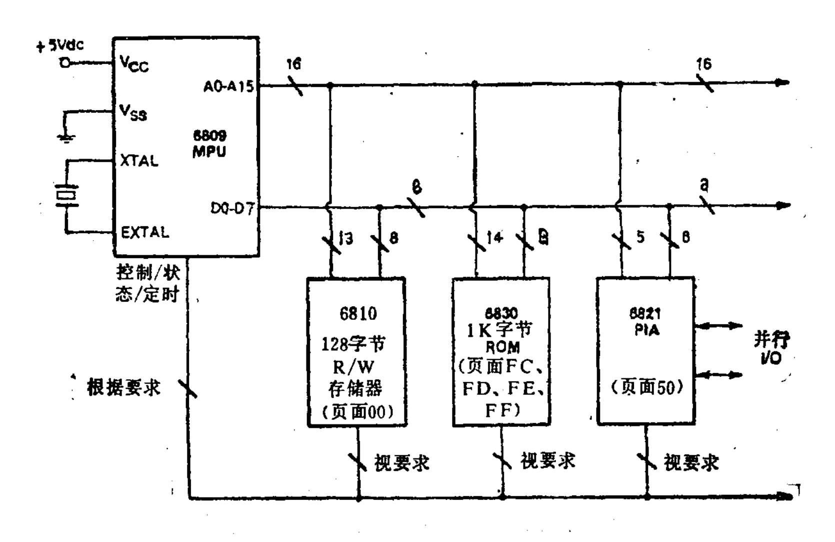


图4.24 6809最小微型计算机系统

图 4.25 所示。从该图中可知：6810读/写存储器在页面零，地址为 $0000～007F；6821 PIA 在页面50，地址为 $5000～$5003；6830ROM 在页面FC、FD、FE和 FF，地址为 $FC00～$FFFF。地址分配中最关键性的问题是对ROM 的安排，因为 6809 的中断向量地址安排在 $FFF0～$FFFF。

PIA设有两个可以连接外部设备的 8 位通道，或称 8 位数据口，它们可以用程序设置为输入口或者输出口，而且每一位都可以单独编程为输入或输出的数据传送口。只要根据外部接口线的要求，就可对 PIA 进行初始化，以后每个口的使用就好象单独的存储器单元一样。这时在6809和 I/O 设备之间（经由PIA）即可以使用6809的装入和存储指令进行数据传送。另外，还可以使用两个PIA口都作为输入或输出，来做16位数据的传送，这时需要在两个PIA口和某个6809的16位寄存器之间，写一点按字节数据长度装入或存储的简单程序即可实现。

现在让我们按6809最小系统进行16位数据通信，举例如下：

图4.25 6809最小系统存储器地址分配

1. 16位数据输入

采用图4.24所示系统和图4.25有关的存储器地址分配关系，写出从16位输入设备使数据输入到累加器D的程序。

假设PIA已经按照两个口（A和B）都作为输入的配置要求进行完初始化。对6800 系统来说，正常接法是PIA的RS 0 （36线）连到地址线A 0 端，RS1（35线）连到地址线A1，此时PIA寄存器地址分配如表4.5所示。但是当PIA接口到6809时，则与以上连接方式相反，R

表 4.5 RS0接A0和RS1接A1的PIA寄存器地址

地　　址	被选PIA寄存器
5000	DDRA或DRB*
5001	CRA
5002	DDRA或DRB*
5003	CRB

* 决定于控制寄存器的第 2 位

S0连A1、RS1连A0，这时PIA寄存器的分配地址如表4.6所示。这样DRA和DRB就被规定到相邻的存储器单元，所以同6809 16位寄存器有关的任何装入或存储指令（LDD、STD、LDX、STX等）都可用其在6809和PIA之间传送16位数据。按照这种思想下述程序将完成上述任务：

```
LDA   #
  50
TFR   A,DPR
LDD   $
00
```

在该程序中，首先把PIA的页面地址号50存入直接页面寄存器。然后就可以使用直接寻址方式把16位数据直接输入到累加器D。 A口输入高位数据字节，B 口输入低位数据字节。

表 4.6 RS0接A1和RS1接A0的PIA寄存器地址

地 址	被选PIA寄存器
5000	DDRA或DRA*
5001	DDRB或DRB*
5002	CRA
5003	CRB

* 决定于控制寄存器的第 2 位

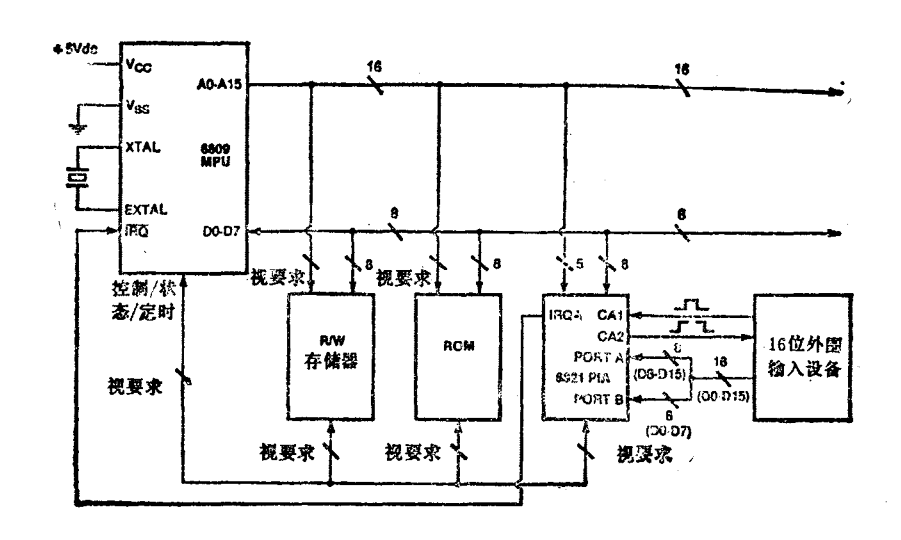
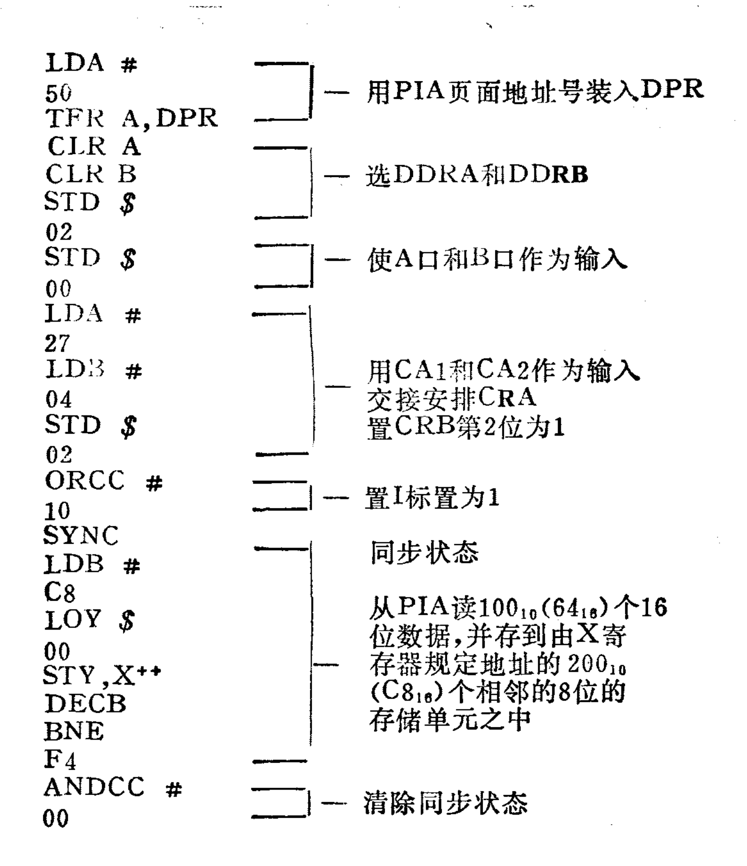


图4.26 16位数据输入同步系统

2. 数据输入同步

设PIA采用如图4.26所示的对外连接方式。要求数据输入操作与外部输入设备实现同步，这样每次外部设备经由PIA的CA1端来中断6809，这时6809把16位数据读出并将其存入X寄存器所指定的存储单元之中。另外，还要求同输入设备建立一套完整的交接过程。所以必须经过PIA的CA2端来确认读出数据操作。知道以上说明之后，试写出 PIA 初始化的程序，以及采用上述同步交接方法实现读出100_{10}(64_{16})个16位数据的程序。该数据输入同步程序如下：



其中RS0、RS1分别接A1和A0，PIA规定地址为5000～5003。首先程序安排PIA，A口控制寄存器CRA的安排是：当CA1为高电平时，对6809产生中断请求IRQ。当按收了中断之后，CA2将变为高电平，而当数据被读出后，立即回到低电平，这样就完成了交接工作。当PIA完成初始化后，条件码寄存器的I位被置1，而且6809用SYNC指令进入同步状态。每次CA1有效工作时，6809都把16位数据读进Y寄存器，并把它们存到由X寄存器规定的两个相邻的存储器单元。在收到另一个$\overline{IRQ}$之前，程序将返回到同步状态。这时读周期将一直重复进行到100_{10}（64_{16}）个16位数据都被存到200_{10}（$C8_{16}$）相邻的8位存储单元中为止。这里应该知道的是完成交接的工作发生在6809经过PIA和输入设备每次传送数据之间。

总之，6809还用在许多16位的多处理器中，如68000 16位微处理器作系统监控工作时，往往使用6809来处理专门的任务。16位数据传送在这种16位系统中将是很普通的事，16位的外部输入设备可以是一个16位处理机。

4.3.2 6809扩充系统

6809是一个很通用高性能的微处理器。由于其软件和硬件都具有较高的性能，在不牺牲性能的情况下，它很易于扩充到第一代和第二代微处理器（象6800、8080等）所不能做到的方面。如对分时工作、高级语言翻译（PASCAL、BASIC、FORTRAN、COBOL）工作都可以有效地进行。

扩充后的6809系统如图4.27所示。其中提供了微型计算机的许多功能，使用了各种6800系列的器件。6843软磁盘控制器（FDC）采用6844DMA控制器进行直接存储器存取。使用6850异步通信接口连接器（ACIA）同CRT显示器、调制解调器（MODEM）和打印机进行串行通信。同时使用6821（PIA）器件作为并行输入输出接口，因此该系统经过PIA接口同16位的68000系统相连接或单独构成系统都很容易实现。所以6809可以组成各种系统，实际上是没有任何限制的。

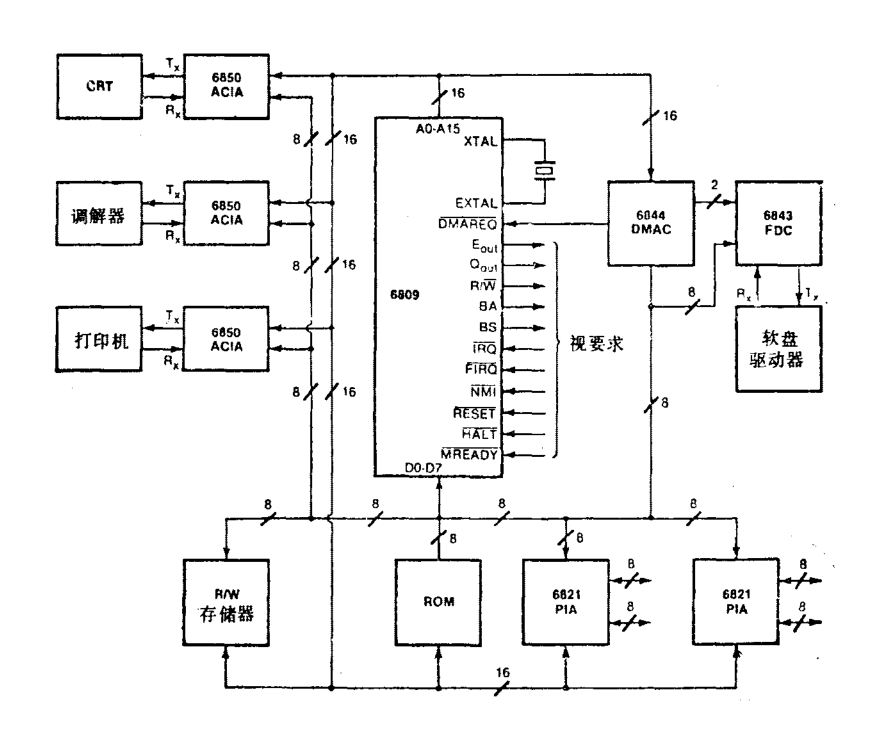


图4.27 扩充后的6809微型计算机系统

4.3.3 6809多处理器系统

多处理器系统是未来的一个发展方向。许多专门的系统功能都可以由多个单独的处理器进行处理，而每个处理器都是拥有自己权限的一个完整的系统。每个单独的处理器系统通常含有专用的读/写存储器、ROM和外部设备。但，每个专用系统又都可以共享一个公用地址和数据总线，而且进入所有系统处理器都可以共享的由全系统存储器和外部设备构成的大系统。在很多情况下，整个系统的操作将由一个16位处理器进行监视，如用MC68000微处理器。系统处理器的任务是协调各专用处理器的工作，差不多就象工程管理主任或项目负责人来协调其它各种人员的工作一样，以期在可能最高效率的情况下完成各项工作。

在以前提过6809E是专门为组成多处理器系统应用而设计出来的微处理器。如前所述，6809E额外设有状态线（LIC和BUSY）它们特别适合于多处理器系统。采用两个6809E的简单的多处理器系统如图4.28所示。其中略去了许多接口的详情，实际系统图是很复杂的，这里只是给出一个多处理器系统的基本概念。

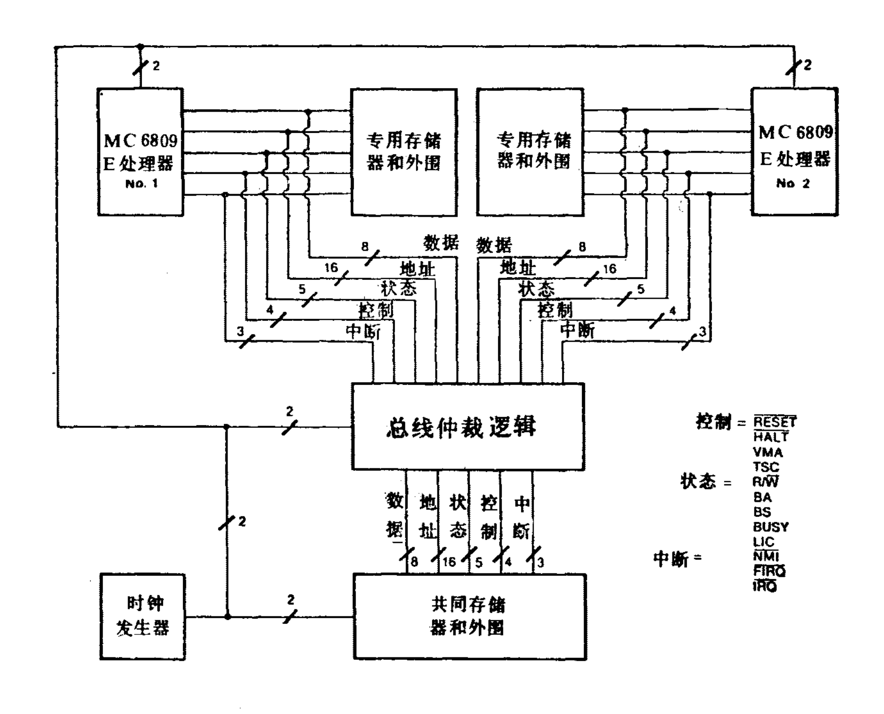


图4.28 6809E多处理器系统

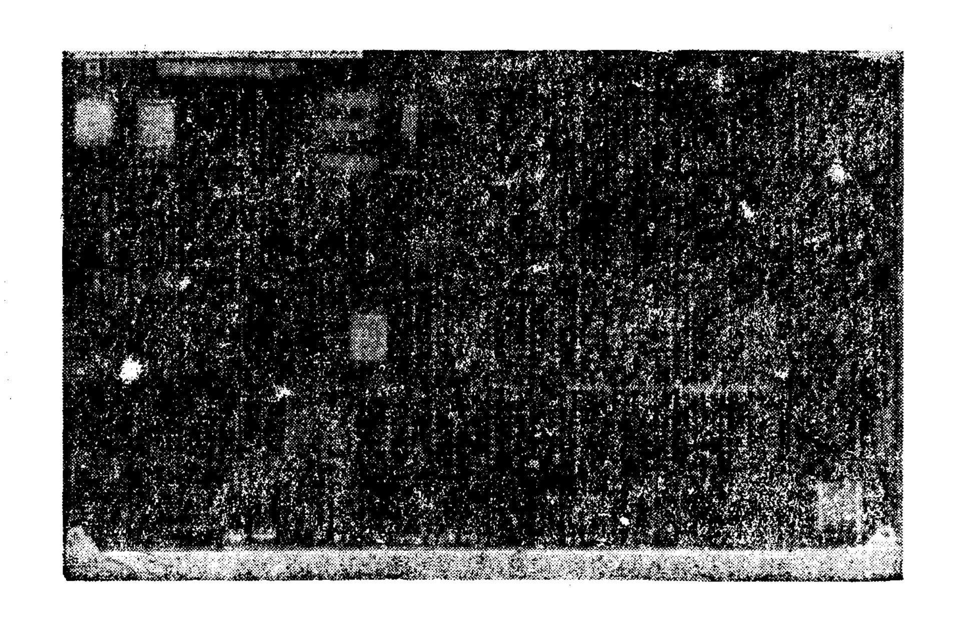
有关6809E组成多处理器系统的实例请见参考文献资料〔2〕。

4.3.4 MEK6809D4单板微型计算机评价系统

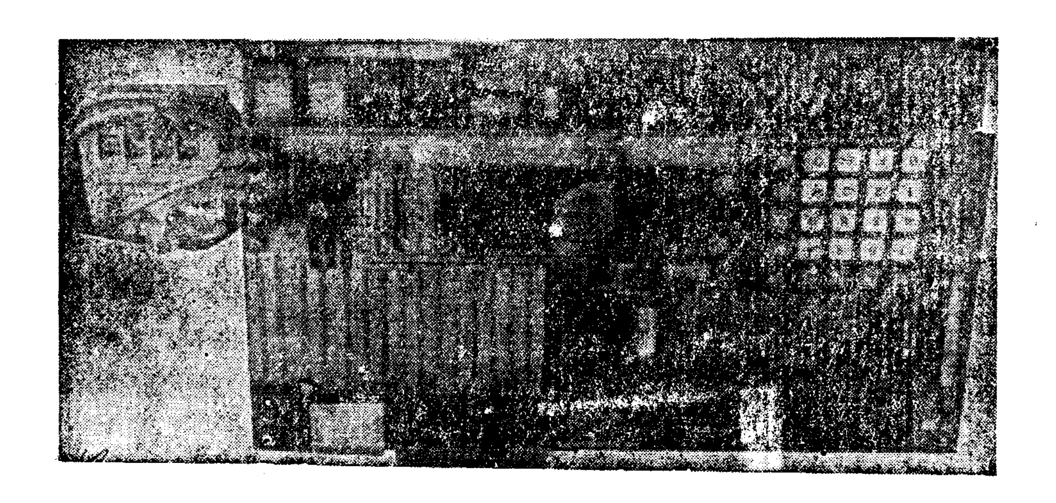
为了了解和掌握6809系统，并为工程应用提供一个评价工具，莫托罗拉公司使用6809研制出一种成本低廉的单板机评价系统。其名称叫MEK6809D4系统，外型结构如图4.29所示。本节主要目的在于对该评价系统做一般性介绍，以对该系统有一个初步了解。详细的资料请阅D4系统的使用参考手册，见参考文献资料〔3〕。

MEK6809D4 设有两种使用方式，称 D4A 和 D4B。D4A 评价系统有两块单板，一块是MEK6809D4微型计算机单板，另一块是MEK68KPD 按键板/电源/显示器单板①。另外，还有墙壁引线的变压器给系统供电，在最小系统配置下，不需要外加电源。D4B型是单板系统，专门使用RS-232C串行终端。在系统中提供完整的RS--232C 接口线路。但用户自己必须外加＋12V、＋5V和－12V电源。本节以介绍D4A型为主。

① 按键板（keypad）和键盘（keyboard）是有区别的，前者电键任意数目，随意安排，由用户定义；后者通常指传统的打字机键盘，有其传统的固定格式和安排。



MEK6809D4



MEK68KPD/MEK6809D4

图4.29 MEK6809D4微型计算机评价系统

上：计算机板；下：按键/电源/显示器板

D4A型单极机的功能安排如图4.30和图4.31所示，其中每一部分功能都标有数字，现对标志数目字的各个部分说明如下：

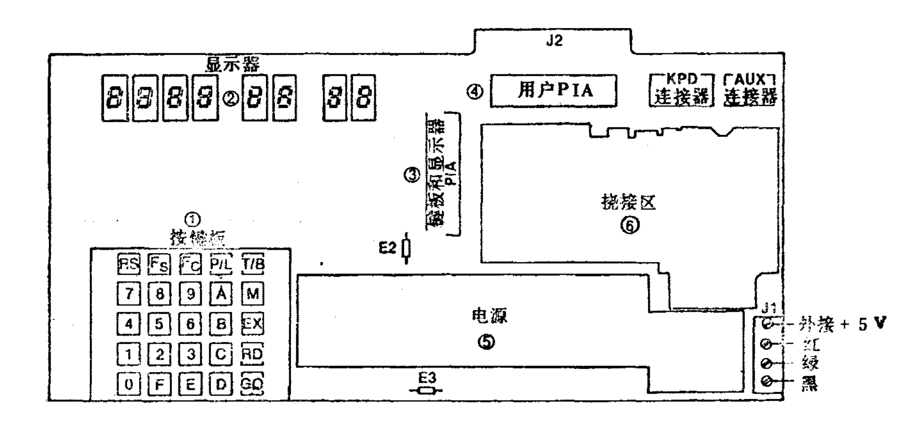


图4.30　MEK68KPD按键板/电源/显示器板

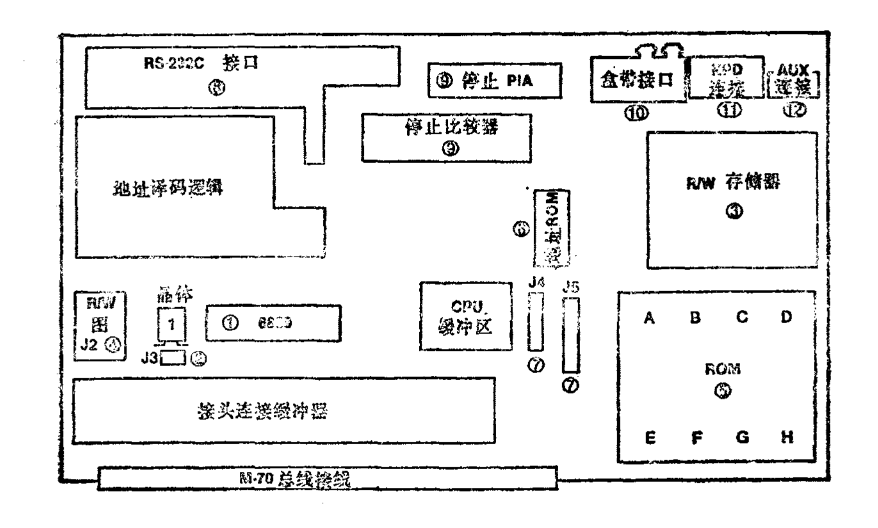


图4.31　MEK6809D4微型计算机板

1.MEK68KPD按键板／电源／显示器板（图4.30）

（1）十六进制按键板——允许操作人员输入十六进制数据（白键）和系统命令（蓝键）。按下某个键时，6809就产生非屏蔽中断$\overline{NMI}$，然后$\overline{NMI}$服务程序检查闭合的键（译码）。所有的键都具有某个功能，见后面介绍。

（2）7段发光二极管显示器（LED）——该部分由8个7段LED显示器组成输出显示。可以显示系统中的内部寄存器、外部存储器和地址信息，另外还可以显示各种使用提示符号，以方便系统使用。8个显示器使用多路切换方式来显示信息内容和多种字符。

（3）按键板/显示器用PIA——该PIA并行接口器件给系统提供两种服务功能：一是按键板译码，二是字符显示。当某个键被按下时，6809产生中断。然后PIA的B口扫描按键板，以确定被按之键在哪一列的位置。按键板数据经A口输入。另外，PIA还兼有显示功能，LED字符标志经A口输出，显示器由B口顺序启动工作（多路切换方式），为防止闪烁，要

以相当快的速率重复显示。规定按键板/显示PIA的地址为$E0E8～$E0FB，其安排如下：

DRA/DDRA	$E0F8
CRA	$E0F9
DRB/DDRB	$E0FA
CRB	$E0FB

（4）用户PIA——该PIA完全是为用户配置而设，A口和B口的所有数据线以及有关的控制线（CA1、CA2、CB1、CB2）都被引出到附近的J2连接插头上。准确的接线位置请参考系统用户手册[3]。规定用户PIA的地址为$E0FC～$E0FF，其安排如下：

DRA/DDRA	$E0FC
CRA	$E0FD
DRB/DDRB	$E0FE
CRB	$E0FF

（5）电源——在MEK68KPD板上设有两个+5V直流电源，一个电源经过调压器VR1驱动8个LED显示器，另一个电源经过调压器VR2供给D4A单板机逻辑使用。外部18V的中心抽头的变压器也经过该板连接器供给系统使用。

特别要提起注意一点是：在板上所设计的电源只能满足最小系统配置的供电要求，如果增加了其它另外的器件或设备，必须外加电源。为了与外接电源连接，首先必须拆去E2和E3的连接电阻器，以防板上的电源调压器损坏。然后再把TTL电平的+5V直流电源接到J1连接器上，如图4.30所示。

（6）绕接区——该区是留给用户把自己所需的外部器件绕接到系统中使用的空白区。为了方便起见，在该区提供了地线和+5V电源二条电源总线。6809的数据、地址、定时、状态和控制信号线都需经KPD和AUX连接器接到微型计算机单板上面。接线的准确规定见系统用户手册。

2. MEK6809D4 微型计算机单板（图4.31）

（1）6809MPU和晶体——该部分是系统的微处理器核心，带有4倍频的晶体振荡器，以产生内部时钟频率。

（2）时钟选择器——跳线器J3可选择内部或外部时钟控制。准确的跳接位置见系统用户手册。

（3）读/写存储器——在板上设有五对2114（1K×4）静态存储器或相当型号的读/写存储器件使用的5K字节的存储器芯片插座。最小系统需使用二片2114（1K字节）芯片，以便作为系统的变量工作区和支持所用的系统堆栈。这1K字节的存储器地址为$E400～$E7FF。增加其它读/写存储器芯片时，其地址可由读/写存储器地址分配跳接器进行选择而定。

（4）读/写存储器地址分配器——地址分配使用跳接器J2，它可以把用户增加的读/写存储器以4K为一块安排在64K字节的内存空间中16种可能的任一区中。准确的跳接位置见系统用户手册。

（5）ROM区——设有8支ROM/EPROM插座，从A到H共有48K字节。在最小系统中配有MCM68332ROM（4K×8）其中装有系统监控程序D4BUG。其它ROM地址的规定都要由ROM地址分配器和ROM种类连接器来确定。通常A～D位置装入容量较大的ROM器件（4K～8K），可作为编辑/汇编程序使用。E～F ROM位置可用较小容量（1K、2K、4K）单电源或三电源的ROM/EPROM器件。ROM插座G留给RS-232C接口所需的专用2K ROM（R2-RS-232）使用，该ROM是MEK6809D4系统按D4B型方式连接串行终端时使用。ROM插座H作为D4A和D4B系统时监控程序D4BUG ROM的4K字节使用。

（6）地址分配ROM——地址分配的ROM，在D4A型和D4B型系统中都需使用，该ROM可使8个ROM器件的地址分配到64K存储空间之中，其地址是唯一的但也是可改变的。该ROM地址分配详细情况见系统用户手册。

（7）ROM类型选择器（J4和J5）——它们是跳接器，对ROM插座A～G可以选择以下任何一种ROM类型：

ROM/EPROM插座A～D

2K×8单电源（MCM2716、TMS2616、MCM68A316E）

4K×8单电源（MCM25A32、TMS2532、MCM68A332）

8K×8单电源（MCM68A764、MCM68A364）

ROM/EPROM插座E～H

1K×8三电源（MCM2708、TMS2708）

2K×8三电源（TMS2716）

2K×8三电源（MCM2716、TMS2516、MCM68A316E）

4K×8三电源（MCM25A32、TMS2532、MCM68A332）

J5连接器安排插座A～F，J4连接器安排插座G～H。在系统中对所用ROM的选择是有极大灵活性的。ROM配置时准确的跳接器位置安排见系统用户手册。

（8）RS-232C接口——按D4B型系统使用时，该部分提供所需RS-232C接口器件（ACIA、波特速率产生器等）。D4A型系统中该区插座可以插串行接口器件。

（9）停止PIA和比较器——D4系统中设有PIA和有关的比较器逻辑来作为暂停在某地址的能力。设置停止地址的目的是使被执行的用户程序做到某个预定的地址时停下来。停止地址被放在停止地址用的PIA中A口和B口。当地址总线上的地址与停止地址相同时，则比较器经过PIA的CA1产生非屏蔽中断$\overline{NMI}$来停止程序的执行。停止PIA自动被初始化，是系统RESET程序的一部分。

（10）盒带机接口——该接口可以作为系统对廉价盒带记录器（录音机）的接口。在D4BUG监控程序中设有录放程序，通过按键板上的P/L和FS键可以提供准确的数据格式和再生功能。数据存储（记录）和再生（放音），使用D4BUG软件，按堪萨斯城标准中300或1200bPS数据格式进行。

（11）KPD连接器——这是24线的连接器，可使微型计算机单板同MEK68KPD板相连。该连接器上设有6809数据线和选择出来的地址线和控制线。加到该连接器上的所有信号都经过缓冲。各引线内容规定见系统用户手册。

（12）AUX连接器——这16条连接器设置目的是可以把所有的6809地址和控制信号加到MEK68KPD上，以便为绕接区使用。加到连接器上的所有信号都进行了缓冲。各种引线

内容含意见系统用户手册。

最后，MK68KPD按键板中各键的功能写在图4.32之中，控制键有以下功能：

- 总清（RESET）整个系统；
- 插入或删除程序中的断点；
- 显示和修改6809内部寄存器内容；
- 检查和修改读/写存储器单元的内容；
- 单步执行程序；
- 计算8位和16位相对地址偏值；
- 在盒带记录器中存储（记录）存储器内容程序；
- 使盒带中内容装入（读出）到存储器；
- 为读/写存储器和ROM提供硬件页面；
- 在不执行总清操作时，从系统程序中退出；
- 选择程序停止地址和在程序停止之前该地址被执行的次数；
- 可定义16个专门的用户功能。

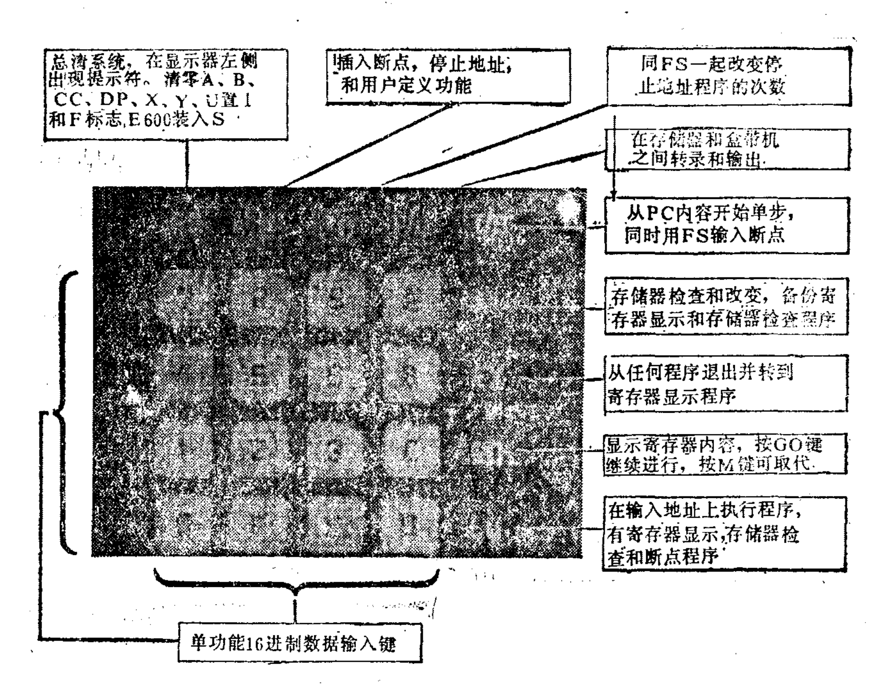


图4.32 D4A型系统按键板电键功能

如何实现上述任务的详细说明，请参考系统用户手册。从以上介绍中可知，D4A是一种非常通用的、功能又很强的微型计算机系统。为了帮助准备使用D4A型系统的用户顺利进行工作，再介绍一些下面的内容。

3. D4A型系统使用要点提示

- 为最小系统方式配置的用户存储器的地址被安排在＄E400～＄E7FF
- 总清（RESET）操作将对程序计数器PC、累加器A、累加器B、X寄存器、Y寄存器、U寄存器和直接页面寄存器DPR进行总清零。另外，条件码寄存器中的F和I标志位被自动置1，而所有其它标志位被置0，即CCR＝50_{16}，同时硬件堆栈指示器S被装入数值为＄E600。

最后总清操作还将消除已经插到用户程序的所有断点。

• 在"GO"命令之后，按电键"EX"时，将会自动地进入寄存器显示程序，而不改变任何内部寄存器的内容。然后按"GO"键时就会继续下一行寄存器显示，如果按"M"键，就要返回显示程序。

• 系统堆栈的起始地址在$E600。如果S寄存器被装入的不是读/写存储器的地址数值，或者没有足够的读/写存储器可供使用时，那么在内部寄存器进行堆栈操作时，显示器将显示："Bad SP? ?"，表明堆栈指示器有问题。

• 系统使用SWI1作为断点程序的中断。

• 为了使用SWI2，SWI2中断服务程序向量地址在$E777：$E778单元。

• 为了使用SWI3，SWI3中断服务程序向量地址在$E775：$E776单元。

• 单步操作（T/B）E标志位置1。

4.应用举例

以下举例可使我们更好掌握D4A系统，同时更加说明6809和D4A单板机有多种用途。

（1）使用SWI2中断

第1步：把以下所示程序装入D4系统，起始地址在$E400单元，参考D4用户手册中程序装入过程一节。

注意：在进入程序后不要按键RS（RESET），可使用退出键EX。

十六进制地址	十六进制内容	记忆符/内容	操作注释
E400	C6	LDB #	主程序
E401	BB	BB	
E402	86	LDA #	
E403	AA	AA	
E404	1F	TFR A,CC	
E405	8B	8B	
E406	10	SWI2	软中断SWI2
E407	3F		
E408	3C	CWAI #	
E409	FF	FF	
•			
•			
E500	CE	LDU #	SWI2中断服务程序
E501	CC	CC	
E502	CC	CC	
E503	3B	RTI	
•			
•			
E777	E5	E5	SWI2中断向量
E778	00	00	

使用SWI2中断服务程序

第2步：按键RD，并使程序计数器放入上述程序的起始地址$E400。

第3步：按键T/B，第一条指令被执行过后，D4系统处在寄存器显示程序。程序计数器内容应是要执行的下一条指令的地址（$E402）。

第 4 步：检验放在累加器B中的数值是否为BB，可按键“GO”二次。这时，如果还想要检查其它寄存器的内容，可以继续按“GO”键。

第 5 步：第二次按T/B键，检验PC的正确内容是否为＄E404，以及累加器A装入的内容是否为AA。

第 6 步：第三次按单步程序T/B键，检查累加器A中数值AA是否被送到直接页面寄存器DP。

第 7 步：再按单步程序键T/B，这时PC的内容为＄FDBC，为什么呢？因为这时恰好执行了软中断SWI2，处理机被指到由SWI2软中断向量所在地址＄FFF4：＄FFF5中给出的地址。这时可以从寄存器显示程序退出按“EX”键，检查放在地址＄FFF4：＄FFF5中SWI2向量是否为＄FDBC。

第 8 步：SWI2指令的执行可使内部寄存器数据进栈。S堆栈指示器在总清操作时即被原始预置到＄E600。这样可以检查＄E600～＄E5F4存储器单元的内容，并验证进栈寄存器的正确内容和进栈的顺序。还要注意一点是当前S寄存器的内容是＄E5F4。查找进栈寄存器内容时应按以下顺序：

S－12→E5F4	CCR
S－11→E5F5	A
S－10→E5F6	B
S－9→E5F7	DPR
S－8→E5F8	X_H
S－7→E5F9	X_L
S－6→E5FA	Y_H
S－5→E5FB	Y_L
S－4→E5FC	U_H
S－3→E5FD	U_L
S－2→E5FE	PC_H
S－1→E5FF	PC_L
S→E600	

第 9 步：现在再按“EX”键和“RD”键回到寄存器显示程序。这时PC的内容仍是ROM区的地址＄FDBC。继续使用单步命令通过该ROM程序，直到PC内容为＄E500地址为止。这里＄E500是SWI2的中断向量。实际上，上面执行ROM程序的目的是使放在地址＄FFF4：＄FFF5中系统的SWI2中断向量＄FDBC转换为我们前面写的程序之中放在地址＄E777：＄E778里的SWI2的中断向量＄E500。如果需要实际了解和学习这种方法时，可以把从＄FDBC地址开始的ROM中的程序指令翻译过来，这样就会了解到以上转换过程是如何实现的。

第10步：在PC内容为＄E500时，单步执行程序，检验执行我们的SWI2中断服务程序是否使U寄存器中装入的数值为＄CCCC

第11步：再用单步执行程序，这是执行RTI指令，所以控制被返回到主程序。检验由RTI指令实现出栈操作的结果，是否已经恢复为原来寄存器的数据内容。

第12步：如果要求用SWI3中断编写程序，为了做到这一点，必须把SWI3的操作码插在地址＄E406：＄E407之中，并在地址＄E775：＄E776中确定所需的中断向量。记住，还要写一个中断服务程序。

（2）掩盖操作码和计算取出向量

第 1 步：在D4系统中装入以下程序，起始地址为＄E400。

十六进制 地址	十六进制 内容	记忆符/内容	操作注释
E400	86	LDA #	装入控制字节
E401	控制字节	CONTROL BYTE	
E402	8E	LDX #	装入向量表的起始地址
E403	FF	FF	
E404	F0	F0	
E405	5F	CLRB	指到控制字节所确定的向量表
E406	10	LBRN	
E407	21		
E408	CB	ADDB #	
E409	02	02	
E40A	44	LSRA	
E40B	24	BCC	
E40C	FB	FB	
E40D	6E	JMP[B,X]	
E40E	95	95	

第 2 步：该程序在3.3.3节计算GO TO转移应用例中提过，可以回顾一下以前对程序的说明。

第 3 步：程序中向量表的起始地址在＄FFF0，这也是6809向量表的起点。这样就可以用控制字节来决定程序将去访问6809的哪个向量。因为6809向量表的地址分配已经固定，所以控制字节将按以下数据来取出6809的向量：

控制字节	取出的向量
00	保　留
02	SWI3
04	SWI2
08	FIRQ
10	IRQ
20	SWI1
40	NMI
80	RESET

第 4 步：在地址＄E401中插入控制字节80，这时就应取出$\overline{\text{RESET}}$中断向量。

第 5 步：执行该程序，那么取出$\overline{\text{RESET}}$操作的结果就会在最左边的显示器中显示出问号（？）。

第 6 步：可以改变控制字节的内容来取出其它的中断向量。在所有情况中，操作的结果都会结束在安排于ROM中各自的中断服务程序上。这时显示器将继续保持空白无显示，或者恢复到寄存器显示程序，这种情况要看取出来的是哪个中断而定。

第 7 步：现在，在读/写存储器中，象3.3.3节计算GO TO程序例那样来建立自己的向量表，同时还要设置某种向量服务程序，以便能够检验向量操作的正确性。可以自行试验一下。另外，也可以在D4A系统中，练习以前提到的各种例题，以便更好地了解和掌握6809微处理器的使用和D4系统的用途。

4.4 6809应用系统

4.4.1 快速中断的应用

6809比6800、8080、Z80这些第二代8位微处理器更适合于大量的数据处理，这一点现在已经是可以理解的了。本节将以6809的快速中断产生用户向量地址的电路，和6809系统中使用标准双密度软磁盘控制器的应用为例进行说明。

1. 用户向量的生成

为了实现快速中断，在如图4.33所示的线路中，产生6809系统用户向量只需再增加一些简单的电路即可。下面简单说明一下工作原理。

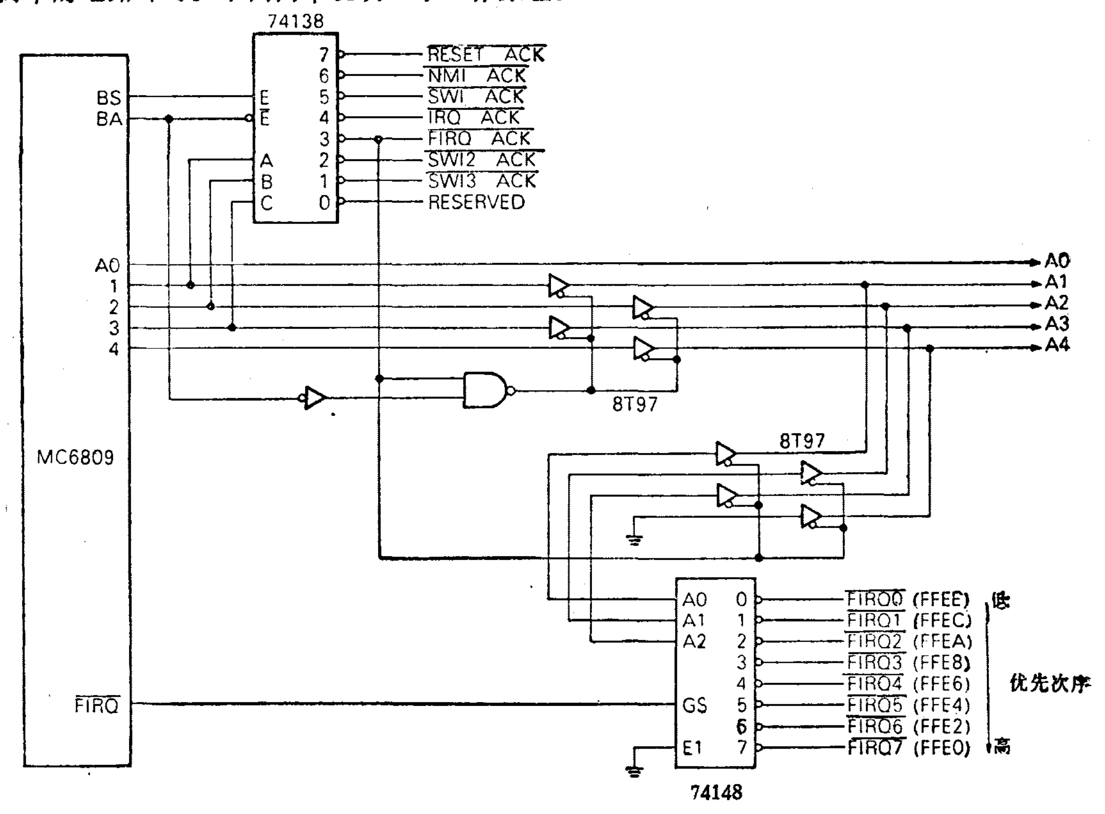


图4.33 6809用户快速中断（FIRQ）向量生成电路

$\overline{FIRQ0}$到$\overline{FIRQ7}$接在外部设备的IRQ输出线上。在8条中断线中，有一条以上产生中断时，则优先级高的数值就会出现在74148的A0、A1、A2各端，而GS端由高电平变为低电平，所以发生了处理器的快速中断信号，在执行中的指令结束的同时，取出中断向量，从而开始中断处理。这时处理器响应后的输出状态是BA＝0、BS＝1。同时在地址线上对各种向量（这里是FIRQ）进行选择。把这些信号如果加到图4.33中的74138电路上，就可以知道取出向量的内容，使用FIRQ的用户向量时，处理器生成的向量是＄FFF6：＄FFF7，除去A0，只使用低4位地址（A1、A2、A3、A4）进行工作，所以移到了＄FFE0：＄FFEF地址之内。这种移动需使A4＝0，使地址线A1、A2、A3分别接在74148的输出端A0、A1、A2

即可实现。关于$\overline{\mathrm{NMI}}$中断的问题，由于$\overline{\mathrm{NMI}}$输入为下降沿触发，所以同时发生两个以上中断时，优先级低的中断则不能执行。本电路仅限在FIRQ、IRQ这两种情况下是有效的。

2. 标准双密度软磁盘数据的传送

6809所设置的快速中断和同步指令（SYNC），使得到目前为止的8位微处理器不可能做到的标准双密度软磁盘系统的软件传送成为可能。双密度软磁盘系统所要求的标准数据传送速度为每16μs传送1字节。

在这种数据传送中，数据的读/写，需要执行指示器（变址寄存器）更新的程序循环。

```
FDC→存储器
LOOP SYNC
     LDA  <FDC
     STA  ,X+
     BRA  LOOP
```

```
存储器→FDC
LOOP SYNC
     LDA  ,X+
     STA  <FDC
     BRA  LOOP
```

6809外同步执行该程序的周期数如下：

SYNC指令执行为2周期，SYNC的触发判别为3周期，采用直接寻地址方式的数据写入或读出为4周期，自动加1变址寻址方式的累加器的读出或者写入为5周期，分支转移指令为3周期，总共为17个机器周期（见3.3节）。

如包含1个周期的安全界限共有18个机器周期，当使用1.125MHz以上工作的6809系统时，一个字节的传送时间相当于在16μs以下，因此，不用DMA方式也可以控制双密度软磁盘系统。

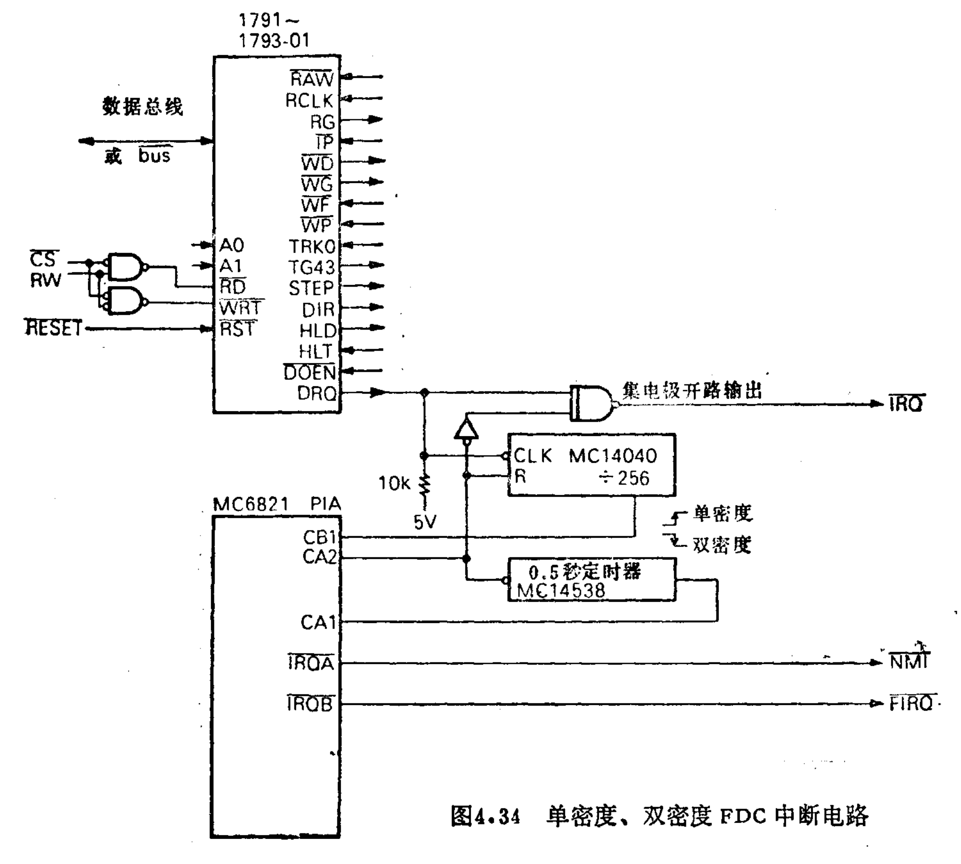


图4.34 单密度、双密度FDC中断电路

在各个扇区进行写入或读出的操作结束时，可以使用快速中断进行CRCC校验等读出纠错的处理。在执行过程中出错，即扇区的读出、写入操作不能进行时，可以使用NMI中断恢复到系统程序之中。

西方数字公司在1791～1793系列中所使用的双密度软磁盘控制器线路如图4.34所示。其中PIA器件作为单元选择，盘片双面的切换、塑料软盘尺寸的切换等条件使用。MC14040是128或256字节/扇区的计数器，作为1791控制器中DRQ（Data Reguest）电路的计数器使用。用上升沿作快速中断请求相当于128字节的单密度扇区；用下降沿作快速中断请求相当于256字节的扇区。IRQ可作为软件屏蔽使用，由作为同步信号用的DRQ信号发出。在数据传送开始之前，CA2为低电平，DRQ有0.5s时间可以工作。在0.5s以内，当扇区的读取或写入还没结束时，可以用$\overline{NMI}$方式从数据传送循环中恢复到系统程序之上。

在该系统进行数据传送过程中，必须完全禁止从其它外部装置产生的中断，由于一个扇区传送结束时要发生快速中断，则需在最后数据之后写入一个字节的无效数据。因此，在移到下一个扇区之前，指示器要减1，所以需要预先把由于无效数据而破坏的数据保留在堆栈之中。

4.4.2 远程数据采集系统

6809微处理器通常还作为自动数据采集系统使用。一种典型的使用6809的数据采集系统如图4.35所示。这种系统可以单独使用，也可以作为大型数据采集系统或分析系统的一部分使用。在许多工业生产制造部门中，为了获得有关产品及其生产加工过程的信息都需采用某

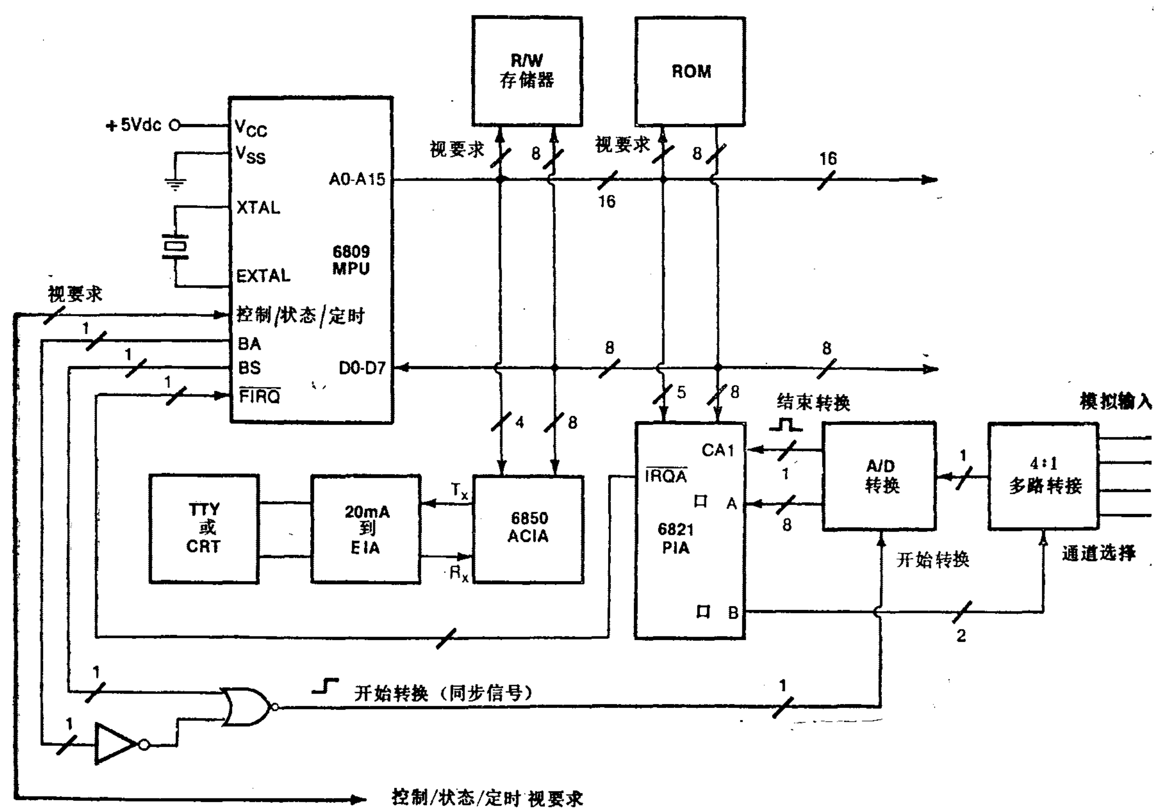


图4.35 远程数据采集系统

种自动数据采集系统。象图4.35中这样一个局部系统可以单独作为产品数据分析系统立即对产品情况作出安排。另外，在某个局部系统上，还可实现进行中过程的简单的统计分析，这样可对生产运行过程给工程技术人员提供出某一瞬间产品和有关工艺的信息。因此，可以保证掌握的当前信息是在不失去控制的工艺过程中得到的，这一点在大量生产制造过程中有特别重要的意义。同时一个局部系统还可以控制工艺变量，为改变产品的性能可以自动地进行补偿调整。许多情况下，几个局部系统可以加入到一个大型多处理机系统中去，进一步提供工程分析、报告和永久性的数据储存。例如，几个这种使用6809的系统可以并入一个大型的68000系统。

现在让我们稍微研究一下图4.35所示的系统。首先，需要提供足够数量的读/写存储器以便进行高速计算和数据暂存。ROM监控程序为数据分析和通信提供所需要的系统操作程序软件。在局部系统ROM中还可以提供高级语言翻译程序，以提高局部工程、数据分析的能力。系统操作员同系统的通信对话可使用电传打字机（TTY）或者CRT数据终端。6809同这些设备的接口可以使用6850ACIA器件，实现串/并变换和建立使用RS-232C（EIA）或20mA电流环的串行通信格式所需要的外部逻辑线路。从外部来的四个模拟量输入端采用4:1的多路转换开关电路实现。通道的选择使用PIA中B口的任意两条数据线作为输出控制线来完成。对四个模拟量输入端的选择，只要使B口有关数据寄存器（DRB）中的某两个数字位准确地写入1或0即可实现。系统中的A/D转换器将使所选中的模拟量信号转换为数字信息。然后把该数字信息加到PIA的A口。这时将有同步程序来控制数据传送。在PIA经常正常地初始化之后，就可以使SYNC指令插在程序之中。当6809执行同步指令SYNC时，BA＝1和BS＝0即表示处在同步状态。这时外部的数字逻辑对BA/BS实行译码给A/D转换器提供一个从低到高的上升沿信号。该信号将通知转换过程开始。一旦模拟信号已被转换成数字信息，A/D转换器就会使PIA中的CA1工作，为6809提供一个有效的$\overline{FIRQ}$中断信号。如果F标志位预先被置1，6809将清除同步状态，而顺序执行下一条程序指令，即读出A口数据。然后分支转移返回到同步指令SYNC，从而为下一个数据字节使过程重复进行。这样就在6809和A/D转换器之间为实现对每一个数据字节的传送而提供了一个完整的交接过程。以上所有软件的细节只要都掌握了解清楚，设计人员就可以毫无困难地编写出控制程序，从而实现上面叙述的怎样安排PIA并得到准确的数据。

第五章　6809实用程序

5.1　6809应用程序

5.1.1　快速付里叶变换(FFT)

快速付里叶变换，最近特别在声音合成输入数据分析和ME（医疗电子仪器）方面发展很快。目前正在试验根据脑电波等的测试波形结果，用FFT进行计算，从其频率数值和波形大小的频谱曲线中，进行疾病的自动诊断。付里叶变换的数学意义及其应用有专门的书籍论述，这里只限于对电子学的含意进行说明。

所有的振荡电信号，其振幅都要随时间发生变化。当把这种时间的函数作成频谱图时，即变换为频率的函数时，这种变换就叫付里叶变换。经过付里叶变换的结果，将是以频率值0（直流）为中心，向$+\infty\sim-\infty$方向展开左右对称的频谱图，见图5.1。

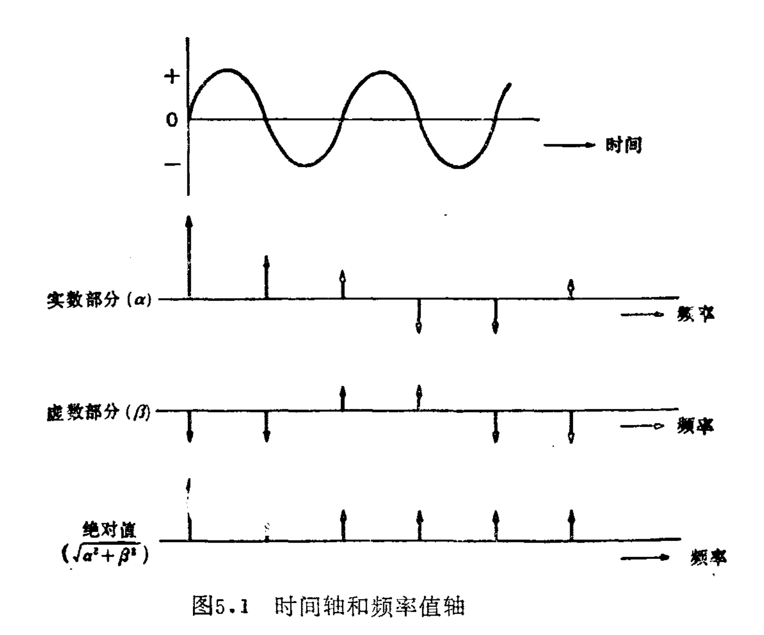


图5.1　时间轴和频率值轴

表5.1所示就是采用四种微处理器进行快速付里叶变换的比较表。表中，6800，8080，MB8861处理器所使用的程序，分别登载在参考文献资料〔17〕，〔18〕，〔19〕之中。

现在，只登载有关MC6809使用的快速付里叶变换的程序清单。在参考文献的程序中，8080的采样频率为128点，只是其它处理器采样频率256点的一半。

设采样点数为N，求出运算处理次数则为$2N\log_2N$，所以采样256点的执行时间必定为128点的16/7倍。但是从128点改为256点的程序执行时间是否为2.3s，这一点是不能说准确

表 5.1 用各种微处理器的快速付里叶变换比较表

处理器 / 项目	MC6809	MC6800	8080	MB8861
采样数目（N）	256	256	128	256
程序字节数（字节）	294	426	524	450
三角函数数据数（字节）	256	256	128	128
工作数据区（字节）	6	34	24	48
概略执行时间（S）	0.6	2.2	1(2、3)*	2
复数运算次数	4096	4096	1792	4096

* 8080使用的程序用128点，总计运算次数为其他处理器的7/16，所以可以认为程序算法经过特别改良之，不到2.3S。

的，这是因为8080工作速度不到其他处理器的4倍，且MC6809具有很强的变址寻址方式和乘法指令（MUL）的原因。MC6800和MB8861的区别在于：MB8861有变址寻址方式的加法指令（ADX）。

现在对使用MC6809的程序说明一下。本程序是用位置独立的技巧进行程序设计的。程序模块有：数位反向顺序化部分；执行数据预处理（CLEAR，MOVE）部分；不做复数运算的第一次通路（PASS1）和做复数运算的第二次以后的通路（FPASS）部分；根据溢出测试和数据比例尺的规格化程序（SCALE）部分；最后是执行2的补码乘法的MPY子程序部分。正弦sin，余弦cos的数据有1/4周期的数据就足够了，但目前本程序使用的是一个周期256字节。

为使程序各个部分实现位置独立化，而且在堆栈内进行数据处理，主要使用变址寻址方式进行。如果需要提高处理速度时，可以变为直接寻址方式，即可加快执行时间，可以提高速度10～20%左右，可在0.5s以内执行完毕。

5.1.2 GPIB 控制器

MC68488通用接口连接器（GPIB）是采用IEEE-STD-488-1978总线标准的、具有发送/接收（通信）功能的、MC6800系列的微处理器外部设备的大规模集成电路芯片。在设计系统终端设备时，不需要控制器功能，只用MC68488即可。但是，在利用MC6809等作CPU的个人计算机或控制器中，则需要增加GPIB控制功能。

在GPIB控制器程序中，是用MC6821（PIA）来处理控制器的功能的。用MC6809的指令编的软件程序约有300字节。更详细的工作原理请参考MOTOROLA公司的应用资料AN-800（英文版）。在AN-800中，是用MC6800指令编写的程序，其基本原理和处理方法都是以MC6800为准的。

表5.2

MC6809使用的快速付里叶变换程序

```
                         NAM    FAST
                         OPT    ABS,LLE=80
                         TTL    FOURIER TRANSFORM SUBROUTINE

                         **
                         * THIS SUBROUTINE PERFORMS A 256
                         * POINT FFT ON THE DATA IN
                         * THE INPUT DATA TABLE.
                         * INPUT DATA IS ASSUMED TO BE TWO'S
                         * COMPLEMENT THE SUBROUTINE GENERATES
                         * A COSINE (REAL) AND SINE (IMAGINARY)
                         * DATA TABLE AT "REAL" AND "IMAG"
                         * THE RESULTANT TRANSFORM DATA IS 128
                         * POINTS SYMMETRICALLY REFLECTED ABOUT
                         * THE CENTER OF THE 256 POINT TABLE.

                         * DATA AREAS

          4000     A INPUT  EQU    $4000
          4500     A REAL   EQU    $4500
          4600     A IMAG   EQU    $4600
          4400     A SINE   EQU    $4400

                         * BASE PAGE PARAMETERS

0020                        ORG    $20
0020      0001     A CELNUM RMB    1            CELLS
0021      0001     A CELCT  RMB    1            CELL COUNTER
0022      0001     A CELDIS RMB    1            CELL OFFSET
0023      0001     A TREAL  RMB    1            TEMP REAL DATA
0024      0001     A TIMAG  RMD    1            TEMP IMAG DATA
0025      0001     A SCLFCT RMB    1            SCALE FACTOR

                         * START OF TRANSFORM

4200                        ORG    $4200

4200 20   00   4202         DRA    START


4202 0F   25       A START  CLR    SCLFCT
                         * INPUT DATA SET-UP

4204 30   8D 03F8  CLEAR    LEAX   IMAG,PCR
4208 5F                     CLRB
4209 6F   80       A CLR1   CLR    ,X+          CLEAR MEMORY
420B 5A                     DECB
420C 26   FB   4209         BNE    CLR1
420E 30   8D 05EE  MOVE     LEAX   INPUT,PCR
4212 31   8D 02EA           LEAY   REAL,PCR
4216 34   04       A        PSHS   B
4218 1F   10       A MOV1   TFR    X,D
421A 86   01       A        LDA    01
421C 54                MOV2 LSRB                REVERSE BIT
421D 49                     ROLA
421E 24   FC   421C         BCC    MOV2
4220 1E   89       A        EXG    A,B
4222 33   AB       A        LEAU   D,Y
4224 E6   80       A        LDB    ,X+          READ INPUT DATA
4226 E7   C4       A        STB    ,U
4228 6A   E4       A        DEC    ,S
422A 26   EC   4218         BNE    MOV1         REPEAT 256 TIMES

                         * FFT FIRST PASS

                         *
                         * SINCE IN PASS 1 ALL ANGLES
                         *   ARE MULTIPLIES OF 180 DEG
                         * THERE ARE NO PRODUCT TERMS.
                         *   AND NO IMAGINARY TERMS YET
                         * HENCE A FAST VERSION OF PASS 1

422C 17   00B2 4301 PASS1   LBSR   SCALE
422F EC   A4       A PA1    LDD    ,Y
4231 34   02       A        PSHS   A
4233 EB   E4       A        ADDB   ,S
4235 E7   A0       A        STB    ,Y+
4237 35   02       A        PULS   A
4239 A0   A4       A        SUBA   ,Y
423B A7   A0       A        STA    ,Y+
423D 6C   E4       A        INC    ,S
423F 2A   EE   422F         BPL    PA1
4241 35   04       A        PULS   B

                         * COMPUTE OF FFT
                         *         PASS 2-N
                         *

4243 86   40       A FPASS  LDA    064
4245 97   20       A        STA    CELNUM       FOR CELL COUNT
4247 86   02       A        LDA    02
4249 97   22       A        STA    CELDIS
424B 17   00B3 4301 NPASS   LBSR   SCALE
424E 96   20       A        LDA    CELNUM       GET NUMBER
4250 97   21       A        STA    CELCT        PUT IN COUNTER
4252 30   8D 02AA           LEAX   REAL,PCR     SET UP POINTER
4256 33   8D 01A6  NCELL    LEAU   SINE,PCR
425A D6   22       A        LDB    CELDIS
425C 34   04       A NC1    PSHS   B
425E 4F                     CLRA
425F D6   22       A        LDB    CELDIS
4261 31   8B       A        LEAY   D,X          Y=X+(CELDIS)
4263 E6   C4       A        LDB    ,U           GET COSINE(U)
4265 34   44       A        PSHS   B,U
4267 1F   30       A        TFR    U,D
4269 CB   40       A        ADDB   #64
426B 1E   30       A        EXG    U,D
426D E6   C4       A        LDB    ,U           GET SINE(U)
426F 35   42       A        PULS   A,U
4271 34   06       A        PSHS   A,B
4273 A6   A4       A        LDA    ,Y           GET RN
4275 E6   E4       A        LDB    ,S           GET COSINE
4277 8D   5E   42D7         BSR    MPY          A=RN*COS
4279 97   23       A        STA    TREAL
427B A6   A4       A        LDA    ,Y           GET RN
427D E6   61       A        LDB    1,S          GET SINE
427F 8D   56   42D7         BSR    MPY          A=RN*SINE
4281 97   24       A        STA    TIMAG
4283 A6   89 0100 A         LDA    IMAG-REAL,X GET IN
4287 34   02       A        PSHS   A            SAVE IN
4289 E6   62       A        LDB    2,S          GET SINE
428B 8D   4A   42D7         BSR    MPY
428D 9B   23       A        ADDA   TREAL
428F 97   23       A        STA    TREAL
4291 35   02       A        PULS   A
4293 E6   E1       A        LDB    ,S++
4295 8D   40   42D7         BSR    MPY
4297 90   24       A        SUBA   TIMAG
4299 97   24       A        STA    TIMAG
429B A6   84       A        LDA    ,X           GET RM
429D 1F   89       A        TFR    A,B
429F 9B   23       A        ADDA   TREAL
42A1 A7   80       A        STA    ,X+
42A3 D0   23       A        SUBB   TREAL
42A5 E7   A4       A        STB    ,Y
42A7 A6   89 00FF A         LDA    IMAG-REAL-1,X
42AB 1F   89       A        TFR    A,B
42AD 9B   24       A        ADDA   TIMAG
42AF A7   89 00FF A         STA    IMAG-REAL-1,X
42B3 D0   24       A        SUBB   TIMAG
42B5 E7   A9 0100 A         STB    IMAG-REAL,Y
42B9 1E   30       A        EXG    U,D
42BB DB   20       A        ADDB   CELNUM
42BD 1E   30       A        EXG    U,D
42BF 35   04       A        PULS   B
42C1 5A                     DECB
42C2 26   98   425C         BNE    NC1
42C4 1F   10       A        TFR    X,D
42C6 DB   22       A        ADDB   CELDIS
42C8 1F   01       A        TFR    D,X
42CA 0A   21       A        DEC    CELCT
42CC 27   29   42F7         BEQ    NP1
42CE 20   86   4256         BRA    NCELL
42D0 0A   21       A        DEC    CELCT
42D2 27   23   42F7         BEQ    NP1          NEXT PASS ?
42D4 16   FF7F 4256         LBRA   NCELL        NO, DO NEXT

                         * 2'S COMP MULTPLY SUBROUTINE

42D7 34   06       A MPY    PSHS   A,B
42D9 A8   61       A        EORA   1,S
42DB 34   01       A        PSHS   CC
42DD 6D   61       A        TST    1,S
42DF 2A   02   42E3         BPL    MPY1
42E1 60   61       A        NEG    1,S
42E3 6D   62       A MPY1   TST    2,S
42E5 2A   02   42E9         BPL    MPY2
42E7 60   62       A        NEG    2,S
42E9 EC   61       A MPY2   LDD    1,S
42EB 3D                     MUL
42EC 35   01       A        PULS   CC
42EE 2A   04   42F4         BPL    MPY3
42F0 C3   0001     A        ADDD   #1
42F3 43                     COMA
42F4 32   62       A MPY3   LEAS   2,S
42F6 39                     RTS

                         * CHANGE PARAMETERS FOR NEXT PASS

42F7 04   20       A NP1    LSR    CELNUM
42F9 27   05   4300         BEQ    DONE         NO MORE CELLS
42FB 08   22       A        LSL    CELDIS
42FD 16   FF4B 424B         LBRA   NPASS        DO NEXT PASS

                         * END OF FFT ROUTINE
```

```
4300 39              DONE   RTS           EXIT FFT

                     *
                     * OVER-RANGE DATA SCALE
                     *

4301 33   8D 01FB    SCALE  LEAU   REAL,PCR
4305 8E   0200     A        LDX    #$200
4308 A6   C0       A SCL2   LDA    ,U+      GET DATA
430A 81   C0       A        CMPA   #$C0     TEST LOWER LIMIT
430C 22   04   4312         BHI    SCL3     SKIP TO NEXT
430E 81   40       A        CMPA   #$40     TEST UPPER LIMIT
4310 24   05   4317         BCC    SCL4
4312 30   1F       A SCL3   LEAX   -1,X     TEST NEXT POINT
4314 26   F2   4308         BNE    SCL2
4316 39                     RTS
4317 0C   25       A SCL4   INC    SCLFCT
4319 33   8D 01E3           LEAU   REAL,PCR SET UP TABLE PTR
431D 8E   0200     A        LDX    #$200
4320 67   C0       A SCL6   ASR    ,U+
4322 30   1F       A        LEAX   -1,X     SCALE NEXT POINT
4324 26   FA   4320         BNE    SCL6
4326 39                     RTS

                     * SINE DATA

4400                        ORG    $4400

4400      7F       A        FCB    $7F,$7F,$7F,$7F
4404      7F       A        FCB    $7F,$7F,$7E,$7E
4408      7D       A        FCB    $7D,$7D,$7C,$7B
440C      7A       A        FCB    $7A,$79,$78,$77
4410      76       A        FCB    $76,$75,$73,$72
4414      71       A        FCB    $71,$6F,$6D,$6C
4418      6A       A        FCB    $6A,$68,$66,$65
441C      63       A        FCB    $63,$61,$5E,$5C
4420      5A       A        FCB    $5A,$58,$56,$53
4424      51       A        FCB    $51,$4E,$4C,$49
4428      47       A        FCB    $47,$44,$41,$3F
442C      3C       A        FCB    $3C,$39,$36,$33
4430      31       A        FCB    $31,$2E,$2B,$28
4434      25       A        FCB    $25,$22,$1F,$1C
4438      19       A        FCB    $19,$16,$12,$0F
443C      0C       A        FCB    $0C,$09,$06,$03
4440      00       A        FCB    $00,$FD,$FA,$F7
4444      F4       A        FCB    $F4,$F1,$EE,$EA
4448      E7       A        FCB    $E7,$E4,$E1,$DE
444C      DB       A        FCB    $DB,$D8,$D5,$D2
4450      CF       A        FCB    $CF,$CD,$CA,$C7
4454      C4       A        FCB    $C4,$C1,$BF,$BC
4458      B9       A        FCB    $B9,$B7,$B4,$B2
445C      AF       A        FCB    $AF,$AD,$AA,$A8
4460      A6       A        FCB    $A6,$A4,$A2,$9F
4464      9D       A        FCB    $9D,$9B,$9A,$98
4468      96       A        FCB    $96,$94,$93,$91
446C      8F       A        FCB    $8F,$8E,$8D,$8B
4470      8A       A        FCB    $8A,$89,$88,$87
4474      87       A        FCB    $87,$85,$84,$83
4478      83       A        FCB    $83,$82,$82,$81
447C      81       A        FCB    $81,$81,$81,$81
4480      81       A        FCB    $81,$81,$81,$81
4484      81       A        FCB    $81,$81,$82,$82
4488      83       A        FCB    $83,$83,$84,$85
448C      86       A        FCB    $86,$87,$88,$89
4490      8A       A        FCB    $8A,$8B,$8D,$8E
4494      8F       A        FCB    $8F,$91,$93,$94
4498      96       A        FCB    $96,$98,$9A,$9B
449C      9D       A        FCB    $9D,$9F,$A2,$A4
44A0      A6       A        FCB    $A6,$A8,$AA,$AD
44A4      AF       A        FCB    $AF,$B2,$B4,$B7
44A8      B9       A        FCB    $B9,$BC,$BF,$C1
44AC      C4       A        FCB    $C4,$C7,$CA,$CD
44B0      CF       A        FCB    $CF,$D2,$D5,$D8
44B4      DB       A        FCB    $DB,$DE,$E1,$E4
44B8      E7       A        FCB    $E7,$EA,$EE,$F1
44BC      F4       A        FCB    $F4,$F7,$FA,$FD
44C0      00       A        FCB    $00,$03,$06,$09
44C4      0C       A        FCB    $0C,$0F,$12,$16
44C8      19       A        FCB    $19,$1C,$1F,$22
44CC      25       A        FCB    $25,$28,$2B,$2E
44D0      31       A        FCB    $31,$33,$36,$39
44D4      3C       A        FCB    $3C,$3F,$41,$44
44D8      47       A        FCB    $47,$49,$4C,$4E
44DC      51       A        FCB    $51,$53,$56,$58
44E0      5A       A        FCB    $5A,$5C,$5E,$61
44E4      63       A        FCB    $63,$65,$66,$68
44E8      6A       A        FCB    $6A,$6C,$6D,$6F
44EC      71       A        FCB    $71,$72,$73,$75
44F0      76       A        FCB    $76,$77,$78,$79
44F4      7A       A        FCB    $7A,$7B,$7C,$7D
44F8      7D       A        FCB    $7D,$7E,$7E,$7F
44FC      7F       A        FCB    $7F,$7F,$7F,$7F

                            END
```

表5.3 GPIB控制器程序

```
00001                           NAM     IEEE

00003                           TTL     GPIB CONTROLLER   MC68488/MC6809

00005                           OPT     ABS,LLE=80
00006                     *
00007                     * MC6809 CONTROLLER ADDRESS = 01
00008                     *

00010                     * SCRATCH AREA

00012A 0000                     ORG     $0
00013A 0000     000A    A LAD    RMB     10        LISTEN ADDRESSES BUFFER
00014A 000A     001E    A RDBFR  RMB     30        DATA READ BUFFER
00015A 0028     0001    A SRQRSP RMB     1         SRQ RESPONSE BYTE
00016A 0029     0002    A TABL   RMB     2

00018                     *
00019                     * SYSTEM STACK AREA
00020                     *

00022A 01FF                     ORG     $1FF

00024A 01FF     0001    A STACK  RMB     1

00026                     *
00027                     * HARDWARE EQUATES
00028                     *

00030A 3000                     ORG     $3000

00032                     * GPIB MC68488 ADDRESS

00034A 3000     0001    A STAMSK RMB     1         0 INT. STATUS/MASK
00035A 3001     0001    A CMDSTA RMB     1         1 COMMAND STATUS
```

```
00036A 3002        0001     A ADDSTM RMB    1          2 ADDR. ST/ADDR.MODE
00037A 3003        0001     A AUXCMD RMB    1          3 AUXILIARY COMMAND
00038A 3004        0001     A ADDRSW RMB    1          4 ADDR.SW / ADDR. REG.
00039A 3005        0001     A SERPOL RMB    1          5 SERIAL POLL
00040A 3006        0001     A CMDPOL RMB    1          6 CMD PASTHRU/PARPOL
00041A 3007        0001     A DTAREG RMB    1          7 DATA REG I/O

00043                         *
00044                         * CONTROL PIA ADDRESS
00045                         *

00047A 3008                          ORG    $3008

00049A 3008        0001     A BUSMST RMB    1          BUS MASTER CONTROL
00050A 3009        0001     A BUSDTB RMB    1          BUS DATA REG.
00051A 300A        0001     A BUSMAC RMB    1
00052A 300B        0001     A BUSDTC RMB    1

00054                         * INITIALIZE GPIA / PIA

00056A 0800                          ORG    $800

00058              0800     A START  EQU    *
00059A 0800 10CE 01FF       A        LDS    #STACK

00061                         * READ IEEE ADDRESS SWITCH ( DIP SW )
00062A 0804 B6     3004     A        LDA    ADDRSW

00064                         * STORE IEEE ADDRESS ON TO MC68488
00065A 0807 B7     3004     A        STA    ADDRSW

00067                         * U-REG SET FOR INITIALIZATION
00068A 080A 33     8C 0B             LEAU   <IZTBL,PCR

00070                         * X-REG = MC68488 START ADDRESS
00071A 080D 8E     3000     A        LDX    #STAMSK
00072A 0810 EC     C1       A IZLOOP LDD    ,U++      READ OFFSET AND DATA
00073A 0812 2B     1E   0832         BMI    EXLOOP    EXIT IF D=$FFFF
00074A 0814 A7     85       A        STA    B,X       STORE INZ DATA
00075A 0816 20     F8   0810         BRA    IZLOOP

00077                         * TABLE FOR MC68488 INZ
00078                         *     BYTE 0   :  DATA TO STORED
00079                         *          1   :  ADDRESS OFFSET
00080                         *     $FFFF    :  END OF DATA

00082A 0818        0003     A IZTBL  FDB    $3,$0,$2,$5,$6,$A,$B,$9
00083A 0828        1F08     A        FDB    $1F08,$140A,$060B,$0608,$FFFF

00085A 0832 4A                EXLOOP DECA
00086A 0833 26     FD   0832         BNE    EXLOOP    DELAY
00087A 0835 86     14       A        LDA    #$14
00088A 0837 B7     3008     A INITDN STA    BUSMST    INITIALIZATION COMPLETE


00090                         *
00091                         *
00092                         * INSERT MAIN PROGRAM HERE
00093                         *
00094                         *


00096                         * TEST PROGRAM

00098              083A     A OUTPUT EQU    *
00099A 083A 8E     10FD     A        LDX    #WRMSG1   DATA OUTPUT TEST
00100A 083D 9F     29       A        STX    TABL      SET UP MSG LOCATION
00101A 083F 86     07       A BET1   LDA    #7
00102A 0841 97     00       A        STA    LAD
00103A 0843 C6     01       A        LDB    #1
00104A 0845 8E     0000     A        LDX    #LAD
```

```
00105A 0848 17      07B5 1000          LBSR   CONNCT    CONNECT MULTIPROG
00106A 084B 9E      29      A          LDX    TABL
00107A 084D C6      07      A          LDB    #7
00108A 084F 17      0827 1079          LBSR   TALKER
00109A 0852 17      08A0 10F5          LBSR   DELAY
00110A 0855 9E      29      A          LDX    TABL
00111A 0857 8C      1115    A          CMPX   #WRMSG4
00112A 085A 27      DE   083A          BEQ    OUTPUT
00113A 085C DC      29      A          LDD    TABL
00114A 085E C3      0008    A          ADDD   #8
00115A 0861 20      DC   083F          BRA    BET1

00117               0863    A INPUT    EQU    *         INPUT TEST
00118A 0863 86      07      A          LDA    #7
00119A 0865 97      00      A          STA    LAD
00120A 0867 C6      01      A          LDB    #1        CONTROLLER TO TALK
00121A 0869 8E      0000               LDX    #LAD
00122A 086C 17      0791 1000          LBSR   CONNCT
00123A 086F 8E      111D    A          LDX    #WRMSGL
00124A 0872 C6      03      A          LDB    #3
00125A 0874 17      0802 1079          LBSR   TALKER
00126A 0877 86      01      A INA1     LDA    #1
00127A 0879 97      00      A          STA    LAD
00128A 087B C6      07      A          LDB    #7
00129A 087D 8E      0000    A          LDX    #LAD
00130A 0880 17      077D 1000          LBSR   CONNCT
00131A 0883 8E      000A    A          LDX    #RDBFR
00132A 0886 17      07BD 1046          LBSR   LISTEN
00133A 0889 4F                         CLRA
00134A 088A 4C                INB      INCA
00135A 088B 2A      FD   088A          BPL    INB
00136A 088D 20      E8   0877          BRA    INA1

00138                         * SRQ TEST

00140A 088F 86      07      A SRQTST   LDA    #7
00141A 0891 97      00      A          STA    LAD
00142A 0893 C6      01      A          LDB    #1
00143A 0895 8E      0000    A          LDX    #LAD
00144A 0898 17      0765 1000          LBSR   CONNCT
00145A 089B 8E      1121    A          LDX    #WRMSGM
00146A 089E C6      03      A          LDB    #3
00147A 08A0 17      07D6 1079          LBSR   TALKER
00148A 08A3 4F                         CLRA
00149A 08A4 4C                SRA1     INCA
00150A 08A5 26      FD   08A4          BNE    SRA1
00151A 08A7 C6      07      A          LDB    #7
00152A 08A9 17      07F5 10A1          LBSR   SRPOLL    GET SRQ STATUS
00153A 08AC C4      40      A          ANDB   #$40
00154A 08AE 27      FE   08AE          BEQ    *         ERROR
00155A 08B0 4F                         CLRA
00156A 08B1 4C                SRA2     INCA
00157A 08B2 26      FD   08B1          BNE    SRA2
00158A 08B4 C6      07      A          LDB    #7
00159A 08B6 17      07E8 10A1          LBSR   SRPOLL
00160A 08B9 5D                         TSTB
00161A 08BA 26      FE   08BA          BNE    *
00162A 08BC 17      0836 10F5          LBSR   DELAY
00163A 08BF 20      CE   088F          BRA    SRQTST

00165A 1000                            ORG    $1000

00167                         * IEEE BUS CONNECT ROUTINE
00168                         * LBSR TO "CONNECT"
00169                         * ENTER WITH B=TALK ADDRESS
00170                         *              X=FWA OF LISTEN ADDR.BUFFER
00171                         *    (MULTIPLE LISTENERS ALLOWED)
```

```
00172                         *    ZERO BYTE IN LAD BFR  = END OF LISTEN ADDRESS.
00173                         * INCLUDE CONTROLLER ADDR WHEN USED FOR I/O

00175A 1000 8D   13    1015 CONNCT BSR    SETPIA
00176A 1002 8D   63    1067        BSR    UNLISN
00177A 1004 CA   40       A        ORB    #$40      OUTPUT TALKER ADDRESS
00178A 1006 1F   98       A        TFR    B,A
00179A 1008 8D   5F    1069 CONN1  BSR    DATSET
00180A 100A A6   80       A        LDA    ,X+
00181A 100C 27   04    1012        BEQ    COND1     ALL LIST ADDR. SENT ?
00182A 100E 8A   20       A        ORA    #$20
00183A 1010 20   F6    1008        BRA    CONN1
00184A 1012 8D   1A    102E COND1  BSR    KESPIA
00185A 1014 39                     RTS

00187                         * SET UP PIA TO TALK ON BUS
00188                         *

00190A 1015 B6   3008     A SETPIA LDA    BUSMST
00191A 1018 84   14       A        ANDA   #$14
00192A 101A 8A   01       A        ORA    #$1       OUTPUT ATN (ATTENTION)
00193A 101C B7   3008     A        STA    BUSMST
00194A 101F 4F              SETPA  CLRA
00195A 1020 B7   300B     A        STA    BUSDTC
00196A 1023 8D   4F    1074        BSR    DATSAV
00197A 1025 86   26       A        LDA    #$26
00198A 1027 B7   300B     A        STA    BUSDTC    PIA SET UP
00199A 102A B6   3009     A        LDA    BUSDTB    CLR PIA STATUS
00200A 102D 39                     RTS


00202                         *
00203                         * RESET PIA TO INPUT
00204                         *

00206A 102E 4F              RESPIA CLRA
00207A 102F B7   300B     A        STA    BUSDTC
00208A 1032 B7   3009     A        STA    BUSDTB
00209A 1035 86   06       A        LDA    #$6
00210A 1037 B7   300B     A        STA    BUSDTC    PIA RESTORED
00211A 103A B6   3007     A        LDA    DTAREG    CLR GPIA DATA REG
00212A 103D B6   3008     A        LDA    BUSMST
00213A 1040 84   14       A        ANDA   #$14
00214A 1042 B7   3008     A        STA    BUSMST    CLR ATN (ATTENTION)
00215A 1045 39                     RTS

00217                         * IEEE BUS CONTROLLER LISTEN ROUTINE
00218                         * LBSR TO "LISTEN"
00219                         * ENTER WITH X=FWA BUFFER AREA  FOR MSG
00220                         *

00222A 1046 B6   3000     A LISTEN LDA    STAMSK    GPIA STATUS
00223A 1049 1F   8A       A        TFR    A,CC
00224A 104B 24   F9    1046        BCC    LISTEN    BI SET ?
00225A 104D 29   0B    105A        BVS    LIST1     YES , EOI ?
00226A 104F B6   3007     A        LDA    DTAREG    NO
00227A 1052 A7   80       A        STA    ,X+
00228A 1054 81   0A       A        CMPA   #$A       INPUT = LF
00229A 1056 27   08    1060        BEQ    LIST2     CR,LF SEQUENCE
00230A 1058 20   EC    1046        BRA    LISTEN

00232A 105A B6   3007     A LIST1  LDA    DTAREG    EOI, READ LAST CHAR
00233A 105D A7   80       A        STA    ,X+
00234A 105F 39                     RTS
```

```
00236A 1060 8D    B3    1015 LIST2  BSR    SETPIA    A C/R , L/F WITHOUT EOI
00237A 1062 8D    03    1067        BSR    UNLISN
00238A 1064 8D    C8    102E        BSR    RESPIA
00239A 1066 39                      RTS

00241                        * HAND SHAKE AND DATA SET
00242                        * TO GPIB DATA LINE   (BUSDTB)

00244A 1067 86    3F       A UNLISN LDA    #$3F
00245A 1069 8D    09    1074 DATSET BSR    DATSAV
00246A 106B B6    300B     A HANDSK LDA    BUSDTC
00247A 106E 2A    FB    106B        BPL    HANDSK
00248A 1070 B6    3009     A        LDA    BUSDTB
00249A 1073 39                      RTS

00251A 1074 43               DATSAV COMA
00252A 1075 B7    3009     A        STA    BUSDTB
00253A 1078 39                      RTS

00255                        *
00256                        *
00257                        * IEEE BUS TALK ROUTINE
00258                        * LBSR TO "TALKER"
00259                        * ENTER WITH B=WORD COUNT-1
00260                        *           X=FWA BUFFER AREA OF MSG
00261                        *

00263A 1079 B6    3000     A TALKER LDA    STAMSK    GPIA STATUS
00264A 107C 84    40       A        ANDA   #$40
00265A 107E 27    F9    1079        BEQ    TALKER    BO SET ?
00266A 1080 A6    80       A        LDA    ,X+       YES OUTPUT CHAR
00267A 1082 B7    3007     A        STA    DTAREG
00268A 1085 5A                      DECB
00269A 1086 26    F1    1079        BNE    TALKER    LAST CHAR ?
00270A 1088 B6    3000     A TALK   LDA    STAMSK    YES
00271A 108B 84    40       A        ANDA   #$40
00272A 108D 27    F9    1088        BEQ    TALK      BO SET ?
00273A 108F 86    20       A        LDA    #$20      YES SET EOI
00274A 1091 B7    3003     A        STA    AUXCMD    AND
00275A 1094 A6    80       A        LDA    ,X+       OUTPUT LAST CHAR
00276A 1096 B7    3007     A        STA    DTAREG
00277A 1099 B6    3000     A TAL1   LDA    STAMSK
00278A 109C 84    40       A        ANDA   #$40
00279A 109E 27    F9    1099        BEQ    TAL1
00280A 10A0 39                      RTS
00281                        *
00282                        * IEEE BUS SERIAL POLL
00283                        * LBSR TO "SRPOLL"
00284                        * ENTER WITH  B= TALKER ADDR FOR SERIAL POLL
00285                        * EXIT WITH B =SRQ STATUS OF TALKER

00287A 10A1 17    FF71  1015 SRPOLL LBSR   SETPIA
00288A 10A4 8D    C1    1067        BSR    UNLISN
00289A 10A6 B6    3004     A        LDA    ADDRSW    CONTROLLER TO LISTEN
00290A 10A9 84    1F       A        ANDA   #$1F
00291A 10AB 8A    20       A        ORA    #$20
00292A 10AD 8D    BA    1069        BSR    DATSET
00293A 10AF CA    40       A        ORB    #$40      OUTPUT TALKER ADDR
00294A 10B1 8D    B6    1069        BSR    DATSET
00295A 10B3 86    18       A        LDA    #$18      SERIAL POLL ENABLE
00296A 10B5 8D    B2    1069        BSR    DATSET
00297A 10B7 17    FF74  102E        LBSR   RESPIA
00298A 10BA B6    3000     A SRLIST LDA    STAMSK    LISTEN FOR RESPONSE
00299A 10BD 1F    8A       A        TFR    A,CC
00300A 10BF 24    F9    10BA        BCC    SRLIST    BI SET ?
00301A 10C1 F6    3007     A        LDB    DTAREG
00302A 10C4 D7    28       A        STB    SRQRSP
```

```
00303A 10C6 17    FF4C 1015        LBSR   SETPIA
00304A 10C9 86    19      A        LDA    #$19
00305A 10CB 8D    9C   1069        BSR    DATSET
00306A 10CD 8D    98   1067        BSR    UNLISN
00307A 10CF 17    FF5C 102E        LBSR   RESPIA
00308A 10D2 39                     RTS

00310                       *
00311                       * IEEE BUS PARALLEL POLL
00312                       *
00313                       * EXIT WITH B=PARALLEL POLL STATUS
00314                       *

00316A 10D3 B6    3008    A PARLEL LDA    BUSMST
00317A 10D6 84    04      A        ANDA   #4          DISABLE BUS DRIVERS
00318A 10D8 8A    09      A        ORA    #9          ENABLE EOI AND ATN
00319A 10DA B7    3008    A        STA    BUSMST
00320A 10DD 4A              PAR1   DECA
00321A 10DE 26    FD   10DD        BNE    PAR1        80 MICROSEC DELAY
00322A 10E0 F6    3009    A        LDB    BUSDTB      READ PARALLEL POLL
00323A 10E3 53                     COMB               MAKE HIGH TRUE
00324A 10E4 B6    3008    A        LDA    BUSMST
00325A 10E7 84    14      A        ANDA   #$14        DROP EOI AND ATN
00326A 10E9 8A    10      A        ORA    #$10        ENABLE BUS DRIVERS
00327A 10EB B7    3008    A        STA    BUSMST
00328A 10EE B6    3008    A        LDA    BUSMST      CLEAR STATUS
00329A 10F1 B6    3009    A        LDA    BUSDTB
00330A 10F4 39                     RTS

00332                       *
00333                       * DELAY ROUTINE
00334                       *

00336A 10F5 8E    7000    A DELAY  LDX    #$7000      250 MS DELAY

00338A 10F8 30    1F      A DEL1   LEAX   -1,X
00339A 10FA 26    FC   10F8        BNE    DEL1
00340A 10FC 39                     RTS

00342A 10FD       4F      A WRMSG1 FCC    8,OT@0001X
00343A 1105       4F      A WRMSG2 FCC    8,OT@0002X
00344A 110D       4F      A WRMSG3 FCC    8,OT@0004X
00345A 1115       4F      A WRMSG4 FCC    8,OT@0010X
00346A 111D       4F      A WRMSGL FCC    4,OT@Z
00347A 1121       4F      A WRMSGM FCC    4,020T

00349                              END
```

5.1.3 Sentronix打印机接口

目前，市场上通行的打印机，大部分都采用Sentronix接口，或者采用以此为准的方式进行。Sentronix接口基本上采用的是三线式交接方式的 8 位并行、字节串行方式，其最高传送速率可达到0.5M字节/秒左右。因为提高了处理器的工作效率，数据传送可高速进行，与打印机的速度无关。

图5.2示出了用Sentronix接口传送一字节数据的过程。图5.3示出了Sentronix 打印机接口电路。表5.4示出打印机接口程序。

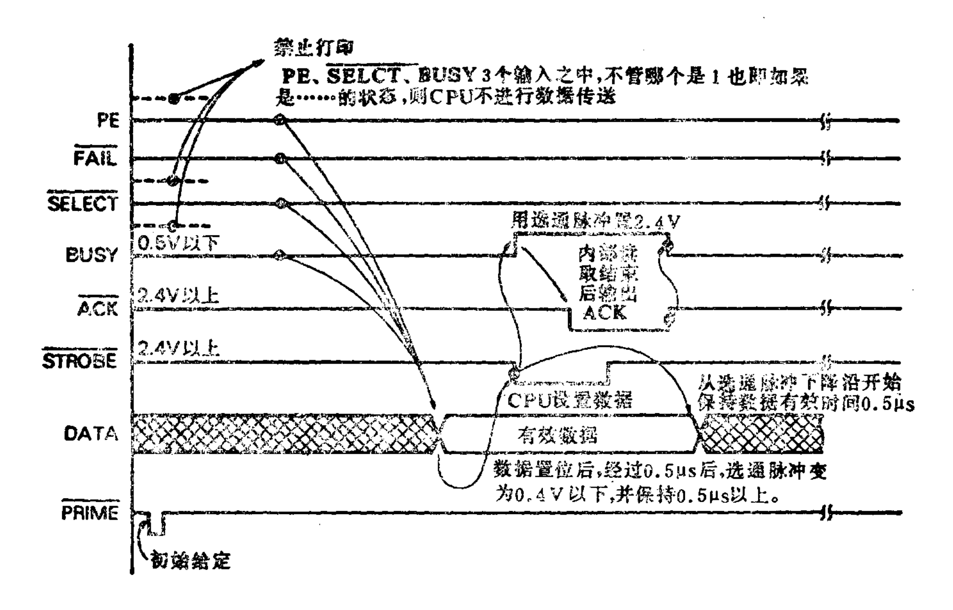


图5.2 用Sentronix接口的一字节数据传送过程

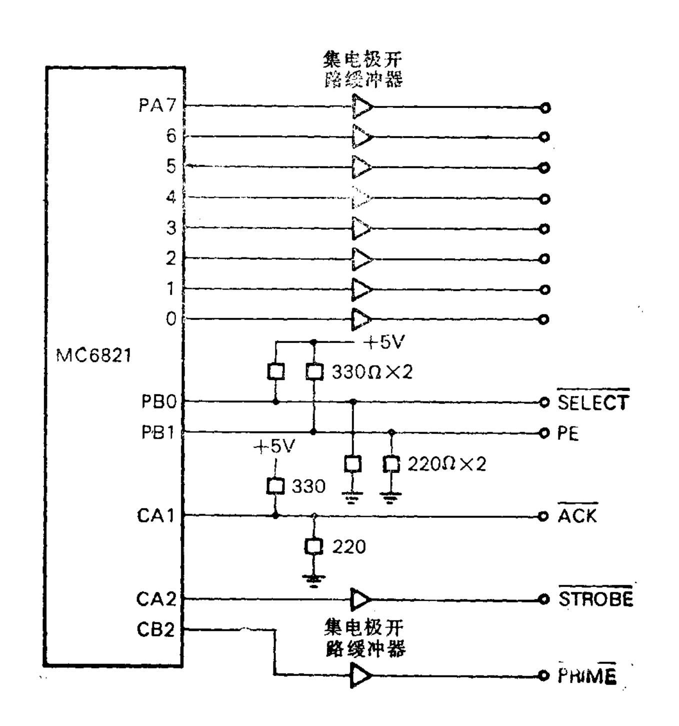


图5.3 Sentronix打印机接口电路

表5.4 SENTRONICS打印机程序

```
00001                          NAM    LPDRV
00002                          TTL    CENTRONICS LINE PRINTER INTERFACE
00003                          OPT    REL,LLE=80

00005                      *
00006                      * LINE PRINTER DRIVER FOR CENTRONICS TYPE
00007                      * INTERFACE THROUGH A PIA WITH OUTPUT
00008                      * CHARACTER ON PORT A, INPUT STATUS ON
00009                      * PORT B
00010                      *

00012                      * PRINTER INTERFACE PIA

00014                      * LOCATION IS DEFINED BY STRD ( START DATA SECTION )
00015                      * PA7-0 TRUE LOGIC PRINT DATA
00016                      * CA1   DATA ACKNOWLEDGE INPUT FROM PRINTER
00017                      * CA2   DATA STROBE OUTPUT TO PRINTER
00018                      * PB0   SELECT INPUT ( NORMAL HIGH )
00019                      * PB1   PAPER END INPUT ( NORMAL LOW )

00021D 0000                    DSCT

00023D 0000       0001    A DATA   RMB    1        PRINT DATA OUTPUT REG
00024D 0001       0001    A CNTRL1 RMB    1        PORT A CONTROL / STATUS
00025D 0002       0001    A STAT   RMB    1        PRINTER STATUS READ
00026D 0003       0001    A CNTRL2 RMB    1        PORT B CONTROL / STATUS

00028P 0000                    PSCT

00030                      * SUBROUTINE TO INITIALIZE PIA

00032             0000    P LPINIT EQU    *
00033P 0000 CC    FF2E    A        LDD    #$FF2E   PORT A = ALL OUTPUT

00035                      * CA2 = AUTO HAND SHAKE

00037P 0003 FD    0000    D        STD    DATA
00038P 0006 86    34      A        LDA    #$34
00039P 0008 B7    0003    D        STA    CNTRL2   SEND PRIM FLAG
00040P 000B 86    3C      A        LDA    #$3C
00041P 000D B7    0003    D        STA    CNTRL2   RESTORE PRIM
00042P 0010 39                     RTS

00044                      * SUBROUTINE TO PRINT CHARACTER FROM ACC A
00045                      * AND CHECK FOR PRINTER ERROR
00046                      * IF ERROR, CARRY IS SET ON RETURN
00047                      *

00049             0011    P LIST   EQU    *
00050P 0011 B7    0000    D        STA    DATA     SEND PRINT DATA

00052                      * DUMMY READ TO GENERATE CA2 DATA STORBE PULSE
00053                      * AND CLEAR CA1 ACKNOWLEDGE PULSE FLAG

00055P 0014 B6    0000    D        LDA    DATA
00056P 0017 34    04      A LIST3  PSHS   B        SAVE ACC B
00057P 0019 F6    0002    D        LDB    STAT     READ PRINTER STATUS
00058P 001C C4    03      A        ANDB   #3
00059P 001E 5A                     DECB            TEST PRINT ERROR
00060P 001F 35    04      A        PULS   B
00061P 0021 26    06   0029        BNE    ERROR    NO PAPER OR NOT SELCTED
00062P 0023 7D    0001    D        TST    CNTRL1
00063P 0026 2A    EF   0017        BPL    LIST3    WAIT ACK PULSE FROM PRINTER
00064P 0028 39                     RTS

00066             0029    P ERROR  EQU    *
00067P 0029 1A    01      A        ORCC   #1       SET CARRY
00068P 002B 39                     RTS
```

续表

```
00070                          * SUBROUTINE TO PRINT STRING AND STRING,CR,LF

00072             002C     P LDATA  EQU    *
00073P 002C 86    0D       A        LDA    #$D      SEND CR
00074P 002E 8D    E1  0011          BSR    LIST
00075P 0030 25    F7  0029          BCS    ERROR
00076P 0032 86    0A       A        LDA    #$A      SEND LF
00077P 0034 8D    DB  0011          BSR    LIST
00078P 0036 25    F1  0029          BCS    ERROR
00079P 0038 20    04  003E          BRA    LDATA1

00081                          * PRINT STRING

00083P 003A 8D    D5  0011 LDATA2 BSR    LIST
00084P 003C 25    EB  0029          BCS    ERROR
00085P 003E A6    80       A LDATA1 LDA    ,X+
00086P 0040 81    04       A        CMPA   #$4      EOT
00087P 0042 26    F6  003A          BNE    LDATA2
00088P 0044 39                      RTS

00090                               END
```

5.1.4 模拟/数字变换

在应用微机的控制系统中，经常遇到的问题是各种检测元件基本上都输出模拟量。因此，用微机进行控制时，一定要把模拟量变为数字量，这一点是不可缺少的。在模拟/数字变换中，如果从大方面分类，有电流开关（MC3408，3410等）型和控制逻辑电路并用的，有把基准电荷加在一定容量电容器上和输入电荷进行比较的电荷平衡型的，有使用向积分电路进行充放电的时间大小的积分型的等等。在这些方式中，它们都各有自己独特的特点。在本节，仅对采用最快速的电流开关型元件进行逐位比较方式(SAR)的硬件和软件加以说明。

硬件是采用MC6821（PIA）和MC3410构成。MC3410是10位数字/模拟（电流）变换用元件，接到PIA的PA1和PA 0 及PB 7 ～PB 0 共10位输入线上。MC3410的电流输出和MC1714的输出，通过满刻度调整电阻的线路，比较其电流数值。输入电流大时，则到PA 7 端的比较器（MLM211）的输出为高电平。如果输入电流低时，则比较器输出为低电平。在SAR变换的过程中，要使输入电流保持不变才行，如果有变化，则需要使用采样和保持电路。这时，可以利用PA 6，PA 5，PA 4 等端进行其它需要的控制。同时，为了满足多输入电路的要求，也可以使用通道切换信号。图5.4示出10位SAR模/数转换电路。

软件程序是由初始化给定的采样程序（INIPIA）和变换子程序（CONY）构成的。因为这两个子程序都是位置独立型的程序，故可以作为主程序中的一部分来使用。变换结果可在累加器D（A，B）中得到。变换周期数为：位数＋1。为了理解变换程序，请参考图5.5。

图5.4　10位SAR模拟/数字变换线路图

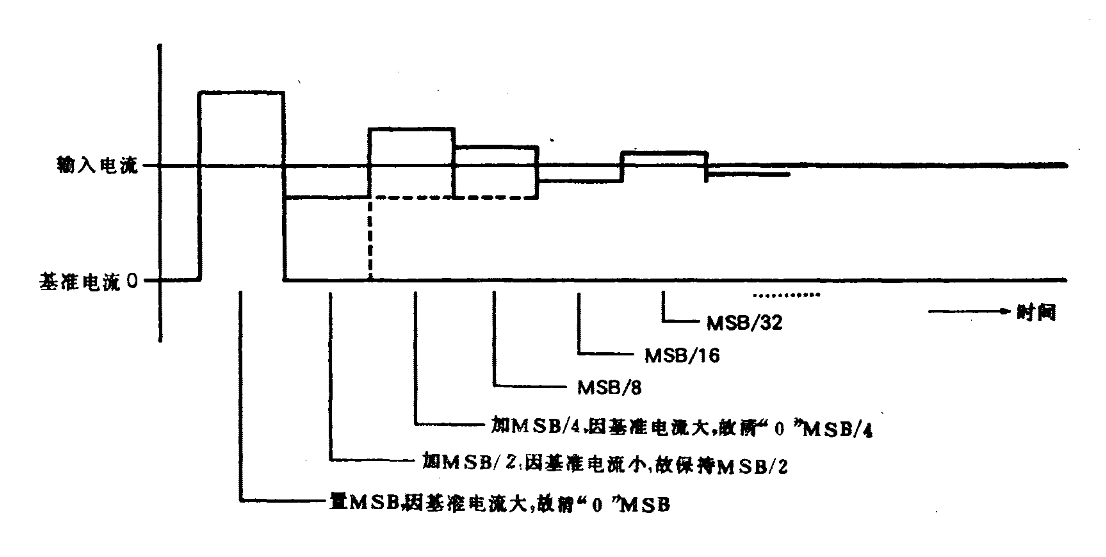


图5.5　比较时间关系图

8，10，12位变换器的不同点在于最起始的MSB的权的大小。它由程序清单中（表5.5）第81行中的累加器D的初始值来给定。变换结束时，被置位的累加器中的内容向右移，一直移到被置位的数字位从16位区域中消失（移到进位位）为止。

表5.5 A/D变换程序

```
00001                          NAM    SAR09
00002                          TTL    8-12 BIT SUCCESSIVE APPROXIMATION A/D

00004                          OPT    REL,LLE=80

00006                      * PIA BIT ASSIGNMENT
00007                      *
00008                      * CA1 CONVERSION TIMING INPUT
00009                      * PA7 COMPARATOR RESULT
00010                      * PA7=1 IF VX < VREF
00011                      *             VREF IS OUTPUT OF DAC

00013                      * DIGITAL TO ANALOG  FOR D/A CHIP
00014                      * PA3 DATA BIT 11 (MSB)
00015                      * PA2 DATA BIT 10
00016                      * PA1 DATA BIT 9
00017                      * PA0 DATA BIT 8
00018                      * PB7 DATA BIT 7
00019                      * PB6 DATA BIT 6
00020                      * PB5 DATA BIT 5
00021                      * PB4 DATA BIT 4
00022                      * PB3 DATA BIT 3
00023                      * PB2 DATA BIT 2
00024                      * PB1 DATA BIT 1
00025                      * PB0 DATA BIT 0 (LSB)

00027                      * PIA ADDRESS

00029D 0000                        DSCT
00030                      * ALLOCATE DATA SECTION FOR PIA

00032                      * PIA  HARDWARE REQUIREMENT

00034                      * PIA RS0  =  A1
00035                      *     RS1  =  A0

00037D 0000      0001    A PIA1AD RMB    1         PORT A DATA
00038D 0001      0001    A PIA1BD RMB    1         PORT B DATA
00039D 0002      0001    A PIA1AC RMB    1         PORT A CTRL/ STAT
00040D 0003      0001    A PIA1BC RMB    1         PORT B CTRL/ STAT

00042                      * PROGRAM START

00044P 0000                        PSCT

00046                      * INITIALIZE PIA REG

00048            0000    P INZPIA EQU    *

00050                      * SELECT DATA DIRECTION REGISTERS

00052P 0000 7F   0002    D         CLR    PIA1AC    SELECT PORT A DDR
00053P 0003 7F   0003    D         CLR    PIA1BC    SELECT PORT B DDR
00054P 0006 CC   7FFF    A         LDD    #$7FFF    PA7=IN ONLY

00056                      * PIA SET UP  -  DATA DIRECTION REG

00058P 0009 FD   0000    D         STD    PIA1AD    SET DDR

00060                      * SELECT DATA REGISTERS

00062P 000C CC   0404    A         LDD    #$0404
00063P 000F FD   0002    D         STD    PIA1AC
00064P 0012 39                     RTS

00066                      * SUBROUTINE  CONVERT A TO D
00067                      * RESULT IN ACC D ON RETURN

00069            0013    P CONV    EQU    *
```

```
00070P 0013 B6   0000    D          LDA    PIA1AD    CLEAR TIMING FLAG
00071P 0016 CC   0000    A          LDD    #0        CLEAR RESULT
00072                     * SAVE RESULT ON TO SYSTEM STACK

00074P 0019 34   06      A          PSHS   D
00075P 001B FD   0000    D          STD    PIA1AD    CLEAR DAC DATA


00077                     * D =  $80 FOR  8 BIT A TO D
00078                     * D = $200 FOR 10 BIT A TO D
00079                     * D = $800 FOR 12 BIT A TO D

00081P 001E CC   0200    A          LDD    #$200     FOR 10 BIT

00083                     * WAIT DATA SET SYNCH CLOCK

00085P 0021 7D   0002    D CONV2    TST    PIA1AC    WAIT CONV TIME
00086P 0024 2A   FB   0021          BPL    CONV2

00088                     * SAVE TEST BIT INFORMATION

00090P 0026 34   06      A CONV1    PSHS   D
00091P 0028 FC   0000    D          LDD    PIA1AD    READ DAC DATA
00092P 002B E3   E4      A          ADDD   ,S        CALCULATE NEW DATA TO DAC
00093P 002D FD   0000    D          STD    PIA1AD    UPDATE DAC
00094P 0030 7D   0000    D          TST    PIA1AD    CLEAR DAC TIMING

00096                     * WAIT DAC TIMING AND TEST DAC RESULT

00098P 0033 7D   0002    D CONV3    TST    PIA1AC    DAC TIMING ?
00099P 0036 2A   FB   0033          BPL    CONV3     WAIT TIMING
00100P 0038 7D   0000    D          TST    PIA1AD    READ COMP RESULT
00101P 003B 2B   0E   004B          BMI    CONV4     BRANCH IF VX < VREF

00103                     * ELSE UPDATE RESULT ON S-STACK AREA

00105P 003D EC   62      A          LDD    2,S       GET PREVIOUS RESULT
00106P 003F E3   E4      A          ADDD   ,S        UPDATE RESULT
00107P 0041 ED   62      A          STD    2,S       SAVE RESULT
00108P 0043 64   E0      A CONV5    LSR    ,S+       SHIFT RIGHT TEST BIT

00110                     * SHIFT RIGHT AND RESTORE STACK POINTER

00112P 0045 66   E0      A          ROR    ,S+
00113P 0047 24   DD   0026          BCC    CONV1     REPT IF C CLEAR
00114P 0049 35   86      A          PULS   D,PC      ELSE RETURN


00116P 004B FC   0000    D CONV4    LDD    PIA1AD    READ DAC BIT PATRN
00117P 004E A3   E4      A          SUBD   ,S        CLEAR PREVIOUS BIT
00118P 0050 20   F1   0043          BRA    CONV5


00120                               END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

5.2 6809系统实用程序

5.2.1 8080仿真程序/调试程序

本程序是用MC6809汇编程序编写的、是运行在MC6809系统中的、由8080仿真程序和调试程序二部分构成的。在程序开始部分约有500字节的调试程序，剩下的约有1.2K字节是仿真程序，总共为1.7K字节的位置独立型程序。仿真程序这一部分由于要求快速性，所以采用直接寻址方式。为此，在装入程序时，要按××00这种形式，即从低地址为00的地方

进行装入。此程序清单中，以地址＄D000作为直接页面地址，而程序是从＄D100地址开始装入的。如果装入的地址不同，例如从＄8100装入时，则直接页面为＄8000，程序的起点自动地变为＄8100地址。

直接页面是指从被装入的××00地址的××中减1的页面（因为仿真/调试程序使用这部分页面），所以应该注意不要和用户程序（8080的机器字）发生冲突。

在MC6809系统中，输入输出的地址包含在存储器地址区中，与此相反，在8080系统中，输入输出的地址包含在I/O地址区中。因此，8080的I/O指令IN，OUT可分配在存储器地址＄E000～＄EDFF的256个字节之中。包含有输入输出子程序的8080程序，在执行的时候，并用同一个外部接口芯片、而能在MC6809总线上工作的情况几乎是没有的。同时，当用8080程序中的子程序时，需要增加和终端之间的通信宏指令作为输入输出指令，因此，向这种终端的输入输出子程序，可换为下面所示的1字节长的宏指令：

＄30　从终端输入一个字符，存入累加器。

＄38　向终端输出一个字符。输出数据为累加器内的数据，用该指令时，累加器的内容不变。

所以8080程序执行时的处理时间，大约需要增加10倍左右的时间。但是，在一般情况下，包括执行输入输出程序时，特别是使用打印机、CRT显示器等串行数据连接线路时，程序的处理能力，就不和处理器本身有关，而是取决于输入输出设备，因此这时就不会感觉到处理器的处理时间变长。实际上，用8080的LISP语言（语言处理程序）执行时，当键盘输入后，几乎没有等待时间，系统的处理能力决定于所用1200波特的终端速度的能力。

为了执行仿真程序，调试程序多少要根据需要加以变动，且因为它主要是作为检查性目的用的，所以功能上没有那么强。

本程序所采用的指令如下：

NE　执行用虚拟程序计数器规定地址的指令。

PC　更新虚拟程序计数器。

RG　显示虚拟寄存器的内容。

GO　依虚拟程序计数器执行，遇到断点时，显示寄存器的内容，返回8080调试程序。同时，遇到HALT指令时，从MC6809系统中的$\overline{\text{Reset}}$处的向量地址返回MC6809系统，不返回直接页面寄存器。

EX　返回MC6809系统，不返回直接页面寄存器。

DK　返回MC6809/MDOS (Motorola Disk Operating System) 系统。(只在EXORCISER EXOR DISK的情况下有效，在其它系统时，作为SWI处理，从SWI向量返回MC6809系统。)

BK　设置中断。

TR　照虚拟程序计数器执行跟踪。在遇到CTRL-W键时，跟踪暂时停止。用其它任何键打入时，则继续下面的步骤。打入断开键时，则打断跟踪。

断开键是为了有CTRL-W的MC6809系统中需要设有CKBRK子程序而用的。CKBRK子程序用断开键使进位位置位，而用CTRL-W时，在CKBRK内循环。用其它键的时候，进位位为清除状态，并返回子程序。不按键时，也要清除进位位并返回。跟踪过程中，忽略断点，遇有HALT指令时，和GO指令处理相同。

ME　是存储器操作，用␣××××，则显示××××地址中的内容，也可以进行修改内容。要改写的数据，应输入二位。输入一位的时候，则原来数据中的低四位移向高四位，而新键入的数据进入低4位。进行数据的修改或者不进行修改时，使用间隔键，则回到前一个地址，用LF(换行)键，则前进到下一个地址。用／键，则可确定更新的数据。以上这些对存储器的操作均用CR（回车）键结束。

没有设置更新虚拟程序计数器以外的寄存器的指令。在更新其它各寄存器内容时，可以加上直接页面起始地址$ 0的偏移值并用ME指令，参考下表进行修改更新：

$20	B	$27	A
$21	C	$28	SP (H)
$22	D	$29	SP (L)
$23	E	$2A	PC (H)
$24	H	$2B	PC (L)
$25	L	$2C	INTE
$26	PSW	$2D,E	BKPT

表5.6的程序清单，是用MC6809宏汇编程序写的8080仿真程序/调试程序清单。标号前的A表示操作是按绝对地址给出的。详细说明可参考附录1 Motorola公司的汇编程序。

另外，因为在程序清单中间，有的是用宏指令编写的，目的码不能从程序清单中全部读出。所以，不了解的宏指令，请参考存储器中机器码的转储清单。

当把该程序移植到其它的6809系统中去时，需要插入下面四个外部子程序：

缺的地址	插入地址	内容
$F015	$D10B, C	键盘来的一个字符输入的子程序，需要返回（LNCH）
$F00F	$D115, 6	是四位十六进制数输入的子程序，从变址寄存器（X）所指的地址按2字节存储数据（INADDR）
$F018	$D110, 1	向终端输出一个字符的子程序（OUTCH）
$F3CD	$D11A, B	检查断点和CTRL-W（CKBRK）。

在各个子程序中，使用直接寻址方式时，则需要对直接页面寄存器进行改变和返回。而且，在监控程序没有给定直接页面寄存器而要使用直接方式时，还要在返回监控程序之前给定DPR数值。故需插入$ D26C地址。

为了检查插入失误，编了一个简易的检查程序，如下面的机器字清单，它是需要用户输入的。

```
5000 20 07 00 00 00 00 00 00 00 33 8C FA 6F C4 6F 41
5010 6F 42 AE 8C ED A6 80 AB 42 A7 42 A6 41 89 00 A7
5020 41 A6 C4 89 00 A7 C4 AC 8C DA 26 E9 39
```

因为该程序可再定位，所以单元的位置与程序结构无关。下面对程序清单进行说明。

计算检查和程序的起始地址为$ 5002，3，结束地址＋1输入到$ 5004，5，请用户从$ 5000单元起动。从$ 5006开始为三字节的检查和。然而，因为程序的最后是RTS指令，故不合适时，请在$ 502C地址处，改为适当的指令。

在MB-6890中，如果执行“EXEC&H5000”型式，计算检查和后，返回BASIC，但不要忘记执行CLEAR指令。

表5.6　8080仿真程序/调试程序源程序清单

执行例

```
*E D100;G
:ME 0000
0000 B8 B8

0001 1A E1

0002 13 CA

0003 B8

:BK P
:BK 000E
:PC 0000
:RG
B C  D E  H L  W  A  SP   PC   INTE BKPT
0000 0000 0000 00 00 0000 0000 00 000E

: 0000 0000 0000 52 00 0000 0001 00 000E
: 0000 0000 E1B8 52 00 0002 0002 00 000E
: 0000 0000 E1B8 52 00 0002 000B 00 000E
: 0000 0000 E1B9 52 00 0002 000C 00 000E
: 0000 0000 E1BA 52 00 0002 000D 00 000E
: 0000 0000 E1BB 52 00 0002 000E 00 000E
: 0000 0000 E1BB 52 00 0002 E1BB 00 000E
:PC 0000
:ME
0000 B8 76
:RG
B C  D E  H L  W  A  SP   PC   INTE BKPT
0000 0000 E1BB 52 00 0002 0000 00 000E

:GO

B C  D E  H L  W  A  SP   PC   INTE BKPT
0000 0000 E1BB 52 00 0002 0000 00 000E
EXBUG09 2.1
```

8080仿真程序/调试程序源程序清单

```
                 NAM    EM8080
                 TTL    DEBUG AND EMULATE PROGRAM   REU 2.0
0000                            ORG    $0

                    * SCRATCH RAM AND SUBROUTINE TABLE

                    * INPUT ONE CHARACTER

0000       0002    A ADRIN  RMB    2

                    * OUTPUT ONE CHARACTER
0002       0002    A ADROUT RMB    2

                    * 8080 REGISTER TOP ADDRESS

0004       0002    A TOPREG RMB    2

0006       0002    A FIRST  RMB    2

                    * BREAK KEY TEST SUBROUTINE ADDRESS

0008       0002    A BRKADR RMB    2

                    * ADDRESS DATA FETCH SUBROUTINE

000A       0002    A AFETCH RMB    2

                    * MEMORY CHANGE OPEN ADDRESS BUFFER

000C       0002    A OPEN   RMB    2
```

```
                    * 8080 PSEUDO REGISTER

000E       0001    A LOC    RMB    1     IN/OUT INSTRUCTION
                     *           ADDRESS HIGH ON MC6809 BUS
0020                        ORG    $20
           0020    A REGS   EQU    *
0020       0001    A REGB   RMB    1
0021       0001    A REGC   RMB    1
0022       0001    A REGD   RMB    1
0023       0001    A REGE   RMB    1
0024       0001    A REGH   RMB    1
0025       0001    A REGL   RMB    1
0026       0001    A PSW    RMB    1
0027       0001    A REGA   RMB    1
0028       0002    A REGSP  RMB    2
002A       0002    A REGPC  RMB    2
002C       0001    A INTE   RMB    1
002D       0002    A BKPT   RMB    2

                    * START DEBUG ROUTINE

D100                        ORG    $D100

D100 20    03    D105 START  BRA    STARTX
D102 12                     NOP
D103 20    2F    D134 SSTART BRA    SOFT

                    * SET UP DP REGISTER

D105 1F    50      A STARTX TFR    PC,D
```

```
D107 4A                   DECA
D108 1F    8B      A      TFR    A,DP

                   * WRITE DEFAULT INCH ADDRESS

D10A 8E    F815    A START1 LDX  #$F815
D10D 9F    00      A        STX  ADRIN

                   * WRITE DEFAULT OUTCH ADDRESS

D10F 8E    F018    A        LDX  #$F018
D112 9F    02      A        STX  ADROUT

                   * WRITE DEFAULT INADDR

D114 8E    F00F    A        LDX  #$F00F
D117 9F    0A      A        STX  AFETCH

                   * WRITE DEFAULT CKBRK

D119 8E    F3CD    A        LDX  #$F3CD
D11C 9F    08      A        STX  BRKADR

                   * SET UP DEFAULT 8080 REGSISTER

D11E C6    20      A        LDB  #REGS
D120 D7    04      A        STD  TOPREG

                   * WRITE DEFAULT 8080 IN/OUT ADDRESS

D122 86    E0      A        LDA  #$E0
D124 97    0E      A        STA  <LOC

                   * CLEAR ALL 8080 REGISTERS

D126 9E    04      A        LDX  TOPREG
D128 C6    26      A        LDB  #38
D12A 6F    80      A CLEAR  CLR  ,X+
D12C 5A                     DECB
D12D 26    FB   D12A        BNE  CLEAR
D12F CE    0003    A        LDU  #3
D132 DF    2A      A        STU  <REGPC

                   * SOFT START
                   * ALL REG AND POINTERS
                   * MUST BE SET FOR PROPER OPERATION

D134 17    0194 D2CB SOFT   LBSR PCRLF
D137 86    3A      A LOOP   LDA  #':
D139 8D    43   D17E        BSR  OUTPUT
D13B 8D    3E   D17B        BSR  INPUT
D13D 81    20      A        CMPA #$20

                   * SPACE KEY FOR NEXT INSTRUCTION

D13F 1027  00C1 D204        LBEQ NEXT
D143 97    06      A        STA  FIRST
D145 8D    34   D17B        BSR  INPUT

                   * CONTROL-X KEY FOR ABORT

D147 81    18      A        CMPA #$18        CNTL-X ?
D149 27    E9   D134        BEQ  SOFT
D14B 97    07      A        STA  FIRST+1
D14D DC    06      A        LDD  FIRST
D14F 30    8D 0052          LEAX CMDEND,PCR
D153 34    10      A        PSHS X
D155 30    8D 0028          LEAX CMD,PCR
D159 10A3  84      A LOOP4  CMPD ,X
D15C 27    0E   D16C        BEQ  LOOP6
D15E 30    04      A        LEAX 4,X         TRY NEXT
D160 AC    E4      A        CMPX ,S
D162 26    F5   D159        BNE  LOOP4
D164 35    10      A        PULS X
D166 86    3F      A        LDA  #'?
D168 8D    14   D17E        BSR  OUTPUT
D16A 20    C8   D134        BRA  SOFT

D16C 32    62      A LOOP6  LEAS 2,S         RESTORE STACK S
D16E 86    20      A        LDA  #'
D170 8D    0C   D17E        BSR  OUTPUT
D172 EC    02      A        LDD  2,X
D174 30    8C 0A            LEAX <CMD,PCR
D177 AD    8B      A        JSR  D,X
D179 20    B9   D134        BRA  SOFT

D17B 16    011E D29C INPUT  LBRA INCH
D17E 16    0127 D2A8 OUTPUT LBRA OUTCH

D181       4E      A CMD    FCC  /NE/
D183       00A1    A        FDB  NE-CMD
D185       50      A        FCC  /PC/
D187       008E    A        FDB  .PC-CMD
D189       52      A        FCC  /RG/
D18B       00EF    A        FDB  RG-CMD
D18D       47      A        FCC  /GO/
D18F       00B0    A        FDB  GO-CMD
D191       45      A        FCC  /EX/
D193       00EB    A        FDB  EXIT-CMD
D195       42      A        FCC  /BK/
D197       00DC    A        FDB  BK-CMD
D199       54      A        FCC  /TR/
D19B       0092    A        FDB  TR-CMD
D19D       4D      A        FCC  'ME'
D19F       8024    A        FDB  MEMORY-CMD
D1A1       44      A        FCC  'DK'
D1A3       00C7    A        FDB  DISK-CMD
           D1A5    A CMDEND EQU  *

                   * MEMORY CHANGE

D1A5 8D    59   D200 MEMORY BSR  OPENAD
D1A7 C6    0C      A MEMO1  LDB  #OPEN
D1A9 1F    B8      A        TFR  DP,A
D1AB 1F    01      A        TFR  D,X
D1AD 17    011B D2CB        LBSR PCRLF
D1B0 17    0110 D2C3        LBSR OUT4HS
D1B3 9E    0C      A MEMO10 LDX  OPEN
D1B5 17    010D D2C5        LBSR OUT2HS
D1B8 17    00E1 D29C MEMO2  LBSR INCH
D1BB 81    0D      A        CMPA #$D
D1BD 26    01   D1C0        BNE  MEMO3
D1BF 39                     RTS
D1C0 81    0A      A MEMO3  CMPA #$A
D1C2 26    08   D1CC        BNE  MEMO4
D1C4 9E    0C      A        LDX  OPEN
D1C6 30    01      A        LEAX 1,X
D1C8 9F    0C      A MEMO30 STX  OPEN
D1CA 20    DB   D1A7        BRA  MEMO1
D1CC 81    2F      A MEMO4  CMPA #'/
D1CE 26    02   D1D2        BNE  MEMO51
D1D0 20    D5   D1A7        BRA  MEMO1
D1D2 81    20      A MEMO51 CMPA #$20
D1D4 26    06   D1DC        BNE  MEMO5
D1D6 9E    0C      A        LDX  OPEN
D1D8 30    1F      A        LEAX -1,X
D1DA 20    EC   D1C8        BRA  MEMO30
D1DC 80    30      A MEMO5  SUBA #$30
D1DE 24    01   D1E1        BCC  MEMO6
D1E0 39                     RTS
D1E1 81    0A      A MEMO6  CMPA #9+1
D1E3 25    0C   D1F1        BCS  MEMOXX
D1E5 81    11      A        CMPA #'A-$30
D1E7 24    01   D1EA        BCC  MEMO7
D1E9 39                     RTS
D1EA 80    07      A MEMO7  SUBA #7
D1EC 81    10      A        CMPA #$10
D1EE 25    01   D1F1        BCS  MEMOXX
D1F0 39                     RTS
D1F1 9E    0C      A MEMOXX LDX  OPEN
D1F3 C6    04      A        LDB  #4
D1F5 68    84      A MEMOX1 LSL  ,X
D1F7 5A                     DECB
D1F8 26    FB   D1F5        BNE  MEMOX1
D1FA AB    84      A        ADDA ,X
D1FC A7    84      A        STA  ,X
D1FE 20    B8   D1B8        BRA  MEMO2

D200 C6    0C      A OPENAD LDB  #OPEN
D202 20    5B   D25F        BRA  BK1XX
D204 8D    1C   D222 NEXT   BSR  NE
D206 16    FF2E D137        LBRA LOOP

D209 30    8D 00C6   MSG    LEAX MSGREG,PCR
D20D 20    7B   D28A        BRA  PDATA

D20F C6    2A      A .PC    LDB  #REGPC
D211 20    4C   D25F        BRA  BK1XX

D213 8D    0D   D222 TR     BSR  NE
D215 34    10      A        PSHS X
D217 8D    05   D21E        BSR  CKBRK
D219 35    10      A        PULS X
D21B 24    F6   D213        BCC  TR
D21D 39                     RTS

D21E 9E    08      A CKBRK  LDX  BRKADR
D220 1F    15      A        TFR  X,PC

D222 DE    2A      A NE     LDU  <REGPC
D224 109E  28      A        LDY  <REGSP
D227 17    01E6 D410        LBSR EXEC
D22A DF    2A      A        STU  <REGPC
D22C 109F  28      A        STY  <REGSP
D22F 20    41   D272        BRA  DISPRG

D231 17    0097 D2CB GO     LBSR PCRLF
D234 DE    2A      A        LDU  <REGPC
D236 109E  28      A        LDY  <REGSP
D239 17    01D4 D410 GO2    LBSR EXEC
D23C 1193  2D      A        CMPU <BKPT
D23F 26    F8   D239        BNE  GO2
D241 DF    2A      A        STU  <REGPC
D243 109F  28      A        STY  <REGSP
D246 20    28   D270        BRA  RG

D248 8E    204C    A DISK   LDX  #$204C
D24B BF    FFF8    A        STX  $FFF8
D24E 8E    F827    A        LDX  #$F827
D251 BF    FFFA    A        STX  $FFFA       SWI VECTOR
D254 8E    F8C3    A        LDX  #$F8C3
D257 BF    FFFC    A        STX  $FFFC       NMI
```

```
D25A 7E   E800     A             JMP    $E800

D25D C6   2D       A BK          LDB    #BKPT
D25F 1F   B8       A BK1XX       TFR    DP,A
D261 1F   01       A             TFR    D,X
D263 109E 0A       A BK2XX       LDY    AFETCH
D266 1F   25       A             TFR    Y,PC

D268 8D   9F    D209 HLT         BSR    MSG
D26A 8D   06    D272             BSR    DISPRG
D26C 6E   9F FFFE A EXIT         JMP    [$FFFE]

D270 8D   97    D209 RG          BSR    MSG
D272 9E   04       A DISPRG      LDX    TOPREG
D274 8D   4D    D2C3             BSR    OUT4HS
D276 8D   4B    D2C3             BSR    OUT4HS
D278 8D   49    D2C3             BSR    OUT4HS
D27A 8D   49    D2C5             BSR    OUT2HS
D27C 8D   47    D2C5             BSR    OUT2HS
D27E 8D   43    D2C3             BSR    OUT4HS
D280 8D   41    D2C3             BSR    OUT4HS
D282 8D   41    D2C5             BSR    OUT2HS
D284 8D   3D    D2C3             BSR    OUT4HS
D286 20   43    D2CB             BRA    PCRLF

                     * PRINT ASCII STRINGS UNTIL EOT

D288 8D   1E    D2A8 PDATAX      BSR    OUTCH
D28A A6   80       A PDATA       LDA    ,X+
D28C 81   04       A             CMPA   #$4
D28E 26   F8    D288             BNE    PDATAX
D290 39                          RTS

                     * INPUT ONE CHARACTER
                     *   8080 MACRO INSTRUCTION = $30

D291 8D   09    D29C .INCH       BSR    INCH
D293 1F   89       A             TFR    A,B
D295 16   029D D535              LBRA   SAVEBC

                     * INPUT ONE CHARACTER FROM ACIA

D298 9E   00       A INCHXX      LDX    ADRIN
D29A 1F   15       A             TFR    X,PC

D29C 34   10       A INCH        PSHS   X
D29E 8D   F8    D298             BSR    INCHXX
D2A0 35   90       A             PULS   X,PC

                     * OUTPUT ONE CHARACTER
                     *      8080 MACRO INSTRUCTION = $38
                     * ROUTE = ACIA / SERIAL DATA LINK
                     *           OR CONSOLE TERMINAL

D2A2 9E   02       A OUTCHX      LDX    ADROUT
D2A4 1F   15       A             TFR    X,PC

D2A6 96   27       A .OUTCH      LDA    <REGA

D2A8 34   10       A OUTCH       PSHS   X
D2AA 8D   F6    D2A2             BSR    OUTCHX
D2AC 35   90       A             PULS   X,PC

D2AE A6   84       A OUT2H       LDA    ,X
D2B0 C6   10       A             LDB    #$10
D2B2 3D                          MUL
D2B3 8D   02    D2B7             BSR    BINCV
D2B5 A6   80       A             LDA    ,X+

D2B7 84   0F       A BINCV       ANDA   #$F
D2B9 8B   30       A             ADDA   #$30
D2BB 81   3A       A             CMPA   #$3A
D2BD 25   E9    D2A8             BCS    OUTCH
D2BF 8B   07       A             ADDA   #$7
D2C1 20   E5    D2A8             BRA    OUTCH

D2C3 8D   E9    D2AE OUT4HS      BSR    OUT2H
D2C5 8D   E7    D2AE OUT2HS      BSR    OUT2H
D2C7 86   20       A PSPACE      LDA    #$20
D2C9 20   DD    D2A8             BRA    OUTCH

D2CB 86   0A       A PCRLF       LDA    #$A
D2CD 8D   D9    D2A8             BSR    OUTCH
D2CF 86   0D       A             LDA    #$D
D2D1 20   D5    D2A8             BRA    OUTCH

D2D3      0D       A MSGREG      FCB    $D,$A
D2D5      42       A             FCC    'B C  D E  H L  W  A
D2EA      53       A             FCC    'SP   PC   INTE BKPT'
D2FD      0D       A             FCB    $D,$A,4

                     *
                     * EMULATION PROGRAM SECTION
                     *

D300      01ED     A ARITH       FDB    ADDX-ARITH,ADCX-ARITH
D304      022C     A             FDB    SUBX-ARITH,SBB-ARITH
D308      026E     A             FI0    ANA-ARITH,XRA-ARITH
D30C      0288     A             FDB    ORAX-ARITH,CMPX-ARITH

D310      0489     A TBL03       FDB    NOP-TBL03,LXIB-TBL03
D314      013D     A             FDB    STAXB-TBL03,INXB-TBL03
D318      0292     A             FDB    INR-TBL03,DCR-TBL03
D31C      0132     A             FDB    MOVI-TBL03,RLC-TBL03
D320      0489     A             FDB    NOP-TBL03,DADB-TBL03
D324      014B     A             FDB    LDAXB-TBL03,DCXB-TBL03
D328      0292     A             FDB    INR-TBL03,DCR-TBL03
D32C      0132     A             FDB    MOVI-TBL03,RRC-TBL03
D330      0489     A             FDB    NOP-TBL03,LXID-TBL03
D334      0144     A             FDB    STAXD-TBL03
D336      02DC     A             FDB    INXD-TBL03,INR-TBL03
D33A      02B3     A             FDB    DCR-TBL03,MOVI-TBL03
D33E      031E     A             FDB    RAL-TBL03,NOP-TBL03
D342      0210     A             FDB    DADD-TBL03,LDAXD-TBL03
D346      02F7     A             FDB    DCXD-TBL03,INR-TBL03
D34A      02B3     A             FDB    DCR-TBL03,MOVI-TBL03
D34E      0326     A             FDB    RAR-TBL03,NOP-TBL03
D352      0167     A             FDB    LXIH-TBL03,SHLD-TBL03
D356      02E4     A             FDB    INXH-TBL03,INR-TBL03
D35A      02B3     A             FDB    DCR-TBL03,MOVI-TBL03
D35E      033A     A             FDB    DAAX-TBL03,NOP-TBL03
D362      0214     A             FDB    DADH-TBL03,LHLD-TBL03
D366      02FF     A             FDB    DCXH-TBL03,INR-TBL03
D36A      02B3     A             FDB    DCR-TBL03,MOVI-TBL03
D36E      0337     A             FDB    CMA-TBL03,.INCH-TBL03
D37%      016E     A             FDB    LXISP-TBL03,STORE-TBL03
D376      02EC     A             FDB    INXSP-TBL03
D378      0292     A             FDB    INR-TBL03,DCR-TBL03
D37C      0132     A             FDB    MOVI-TBL03,STC-TBL03
D380      FF96     A             FDB    .OUTCH-TBL03
D382      0218     A             FDB    DADSP-TBL03
D384      01A5     A             FDB    LOAD-TBL03,DCXSP-TBL03
D388      0292     A             FDB    INR-TBL03,DCR-TBL03
D38C      0132     A             FDB    MOVI-TBL03,CMC-TBL03

D390      0390     A TBLCF       FDB    RNZ-TBLCF
D392      03F4     A             FDB    POPB-TBLCF,JNZ-TBLCF
D396      02D2     A             FDB    JUMP-TBLCF,CNZ-TBLCF
D39A      03D8     A             FDB    PUSHB-TBLCF,ADI-TBLCF
D39E      031F     A             FDB    RST-TBLCF
D3A0      0389     A             FDB    RZ-TBLCF,RET-TBLCF
D3A4      02EF     A             FDB    JZ-TBLCF,JUMP-TBLCF
D3A8      0344     A             FDB    CZ-TBLCF,CALL-TBLCF
D3AC      0177     A             FDB    ACI-TBLCF,RST-TBLCF
D3B0      0382     A             FDB    RNC-TBLCF,POPD-TBLCF
D3B4      02E7     A             FDB    JNC-TBLCF,OUT-TBLCF
D3B8      033C     A             FDB    CNC-TBLCF,PUSHD-TBLCF
D3BC      01C6     A             FDB    SUI-TBLCF,RST-TBLCF
D3C0      037B     A             FDB    RC-TBLCF,RET-TBLCF
D3C4      02DF     A             FDB    JC-TBLCF,IN-TBLCF
D3C8      0334     A             FDB    CCALL-TBLCF,CALL-TBLCF
D3CC      01DA     A             FDB    SBI-TBLCF,RST-TBLCF
D3D0      03AC     A             FDB    RPO-TBLCF,POPH-TBLCF
D3D4      0317     A             FDB    JPO-TBLCF,XTHL-TBLCF
D3D8      036C     A             FDB    CPO-TBLCF,PUSHH-TBLCF
D3DC      01E7     A             FDB    ANI-TBLCF,RST-TBLCF
D3E0      03A5     A             FDB    RPE-TBLCF,PCHL-TBLCF
D3E4      030F     A             FDB    JPE-TBLCF,XCHG-TBLCF
D3E8      0364     A             FDB    CPE-TBLCF,CALL-TBLCF
D3EC      01F4     A             FDB    XRI-TBLCF,RST-TBLCF
D3F0      0397     A             FDB    RP-TBLCF,POPW-TBLCF
D3F4      02FF     A             FDB    JP-TBLCF,DI-TBLCF
D3F8      0354     A             FDB    CP-TBLCF,PUSHW-TBLCF
D3FC      01FF     A             FDB    ORI-TBLCF,RST-TBLCF
D400      039E     A             FDB    RM-TBLCF,SPHL-TBLCF
D404      0307     A             FDB    JM-TBLCF,EI-TBLCF
D408      035C     A             FDB    CM-TBLCF
D40A      0326     A             FDB    CALL-TBLCF,CPI-TBLCF
D40E      031F     A             FDB    RST-TBLCF

                     * EMULATION EXECUTE

D410 E6   C0       A EXEC        LDB    ,U+
D412 2B   17    D42B             BMI    GROUP8
D414 C5   40       A             BITB   #$40
D416 27   08    D420             BEQ    GRP03

                     * OPCODE $76 IS BREAK POINT
                     * EXECUTION IS ABORTED BY BKPT

D418 C1   76       A             CMPB   #$76
D41A 1027 FE4A D268              LBEQ   HLT
D41E 20   77    D497             BRA    MOVE

D420 30   8D FEEC    GRP03       LEAX   TBL03,PCR
D424 4F              GRPTBL      CLRA
D425 58                          LSLB
D426 49                          ROLA
D427 EC   8B       A             LDD    D,X
D429 6E   8B       A             JMP    D,X

D42B C5   40       A GROUP8      BITB   #$40
D42D 26   0B    D43A             BNE    GRPCF
D42F 54                          LSRB
D430 54                          LSRB
D431 54                          LSRB
D432 C4   07       A             ANDB   #$7
D434 30   8D FEC8                LEAX   ARITH,PCR
D438 20   EA    D424             BRA    GRPTBL
D43A 30   8D FF52    GRPCF       LEAX   TBLCF,PCR
D43E C4   3F       A             ANDB   #$3F
D440 20   E2    D424             BRA    GRPTBL
```

```
D442 E6   C4        A MOUI   LDB    ,U
D444 8D   53     D499        BSR    SAVED
D446 33   41        A        LEAU   1,U
D448 39                      RTS

D449 109E 24        A SPHL   LDY    REGH
D44C 39                      RTS

D44D 9E   20        A STAXB  LDX    REGB
D44F 96   27        A        LDA    REGA
D451 A7   84        A        STA    ,X
D453 39                      RTS

D454 9E   22        A STAXD  LDX    REGD
D456 96   27        A        LDA    REGA
D458 A7   84        A        STA    ,X
D45A 39                      RTS

D45B 9E   20        A LDAXB  LDX    REGB
D45D A6   84        A        LDA    ,X
D45F 97   27        A        STA    REGA
D461 39                      RTS

D462 9E   22        A LDAXD  LDX    REGD
D464 A6   84        A        LDA    ,X
D466 97   27        A        STA    REGA
D468 39                      RTS

D469 EC   C1        A LXIB   LDD    ,U++
D46B 1E   89        A        EXG    A,B
D46D DD   20        A        STD    REGB
D46F 39                      RTS
D470 EC   C1        A LXID   LDD    ,U++
D472 1E   89        A        EXG    A,B
D474 DD   22        A        STD    REGD
D476 39                      RTS

D477 EC   C1        A LXIH   LDD    ,U++
D479 1E   89        A        EXG    A,B
D47B DD   24        A        STD    REGH
D47D 39                      RTS

D47E EC   C1        A LXISP  LDD    ,U++
D480 1E   89        A        EXG    A,B
D482 1F   02        A        TFR    D,Y
D484 39                      RTS

D485 A6   5F        A GETSU  LDA    -1,U
D487 84   07        A        ANDA   #$7
D489 81   06        A        CMPA   #%110
D48B 26   05     D492        BNE    GETS1
D48D 9E   24        A        LDX    REGH
D48F E6   84        A        LDB    ,X
D491 39                      RTS
D492 9E   04        A GETS1  LDX    <TOPREG
D494 E6   86        A        LDB    A,X
D496 39                      RTS

                      * MOVE INSTRUCTION

D497 8D   EC     D485 MOVE   BSR    GETSU
D499 A6   5F        A SAVED  LDA    -1,U
D49B 44                      LSRA
D49C 44                      LSRA
D49D 44                      LSRA
D49E 84   07        A        ANDA   #$7
D4A0 81   06        A        CMPA   #%110
D4A2 26   05     D4A9        BNE    SAVDR
D4A4 9E   24        A        LDX    REGH
D4A6 E7   84        A        STB    ,X
D4A8 39                      RTS
D4A9 9E   04        A SAVDR  LDX    <TOPREG
D4AB E7   86        A        STB    A,X
D4AD 39                      RTS

D4AE 8D   27     D4D7 STORE  BSR    GETADR
D4B0 96   27        A        LDA    REGA
D4B2 A7   84        A        STA    ,X
D4B4 39                      RTS

D4B5 8D   20     D4D7 LOAD   BSR    GETADR
D4B7 E6   84        A        LDB    ,X
D4B9 D7   27        A        STB    REGA
D4BB 39                      RTS

D4BC 8D   19     D4D7 SHLD   BSR    GETADR
D4BE DC   24        A        LDD    REGH
D4C0 1E   89        A        EXG    A,B
D4C2 ED   84        A        STD    ,X
D4C4 39                      RTS

D4C5 8D   10     D4D7 LHLD   BSR    GETADR
D4C7 EC   84        A        LDD    ,X
D4C9 1E   89        A        EXG    A,B
D4CB DD   24        A        STD    REGH
D4CD 39                      RTS

D4CE DC   22        A XCHG   LDD    REGD
D4D0 9E   24        A        LDX    REGH
D4D2 9F   22        A        STX    REGD
D4D4 DD   24        A        STD    REGH
D4D6 39                      RTS

D4D7 EC   C1        A GETADR LDD    ,U++
```

```
D4D9 1E   89        A        EXG    A,B
D4DB 1F   01        A        TFR    D,X
D4DD 39                      RTS

D4DE DC   24        A XTHL   LDD    REGH
D4E0 1E   89        A        EXG    A,B
D4E2 AE   A4        A        LDX    ,Y
D4E4 ED   A4        A        STD    ,Y
D4E6 1F   10        A        TFR    X,D
D4E8 1E   89        A        EXG    A,B
D4EA DD   24        A        STD    REGH
D4EC 39                      RTS
D4EF DB   27        A        ADDB   REGA
D4F1 20   42     D535        BRA    SAVEBC

D4F3 D6   27        A ADI    LDB    REGA
D4F5 EB   C0        A        ADDB   ,U+
D4F7 20   3C     D535        BRA    SAVEBC

D4F9 8D   8A     D485 ADCX   BSR    GETSU
D4FB DB   27        A ADCXX  ADDB   REGA
D4FD D7   27        A        STB    REGA
D4FF D6   26        A        LDB    PSW
D501 C4   01        A        ANDB   #$1
D503 D9   27        A        ADCB   REGA
D505 20   2E     D535        BRA    SAVEBC

D507 E6   C0        A ACI    LDB    ,U+
D509 20   F0     D4FB        BRA    ADCXX

D50B DC   20        A DADB   LDD    REGB
D50D D3   24        A DADXX  ADDD   REGH
D50F DD   24        A        STD    REGH
D511 1F   A8        A        TFR    CC,A
D513 84   01        A        ANDA   #$1
D515 D6   26        A        LDB    PSW
D517 C4   FE        A        ANDB   #$FE
D519 D7   26        A        STB    PSW
D51B 9A   26        A        ORA    PSW
D51D 97   26        A        STA    PSW
D51F 39                      RTS

D520 DC   22        A DADD   LDD    REGD
D522 20   E9     D50D        BRA    DADXX

D524 DC   24        A DADH   LDD    REGH
D526 20   E5     D50D        BRA    DADXX
D528 1F   20        A DADSP  TFR    Y,D
D52A 20   E1     D50D        BRA    DADXX

D52C 17   FF56   D485 SUBX   LBSR   GETSU
D52F 34   04        A        PSHS   B
D531 D6   27        A        LDB    REGA
D533 E0   E0        A        SUBB   ,S+

D535 D7   27        A SAVEBC STB    REGA
D537 1F   A8        A SAVECC TFR    CC,A
D539 34   02                 PSHA
D53B 1F   89        A        TFR    A,B
D53D 84   01        A        ANDA   #1          CARRY
D53F 8A   02        A        ORA    #2          ALLWAYS 1
D541 97   26        A        STA    PSW
D543 54                      LSRB
D544 C4   10        A        ANDB   #$10        HALF CARRY?
D546 DA   26        A        ORB    PSW
D548 D7   26        A        STB    PSW
D54A 35   02                 PULA
D54C C6   10        A        LDB    #$10
D54E 3D                      MUL
D54F C4   C0        A        ANDB   #$C0
D551 DA   26        A        ORB    PSW
D553 D7   26        A        STB    PSW
D555 39                      RTS


D556 D6   27        A SUI    LDB    REGA
D558 E0   C0        A        SUBB   ,U+
D55A 20   D9     D535        BRA    SAVEBC

D55C 17   FF26   D485 SBB    LBSR   GETSU
D55F 34   04        A SBBX   PSHS   B
D561 96   26        A        LDA    PSW
D563 44                      LSRA
D564 D6   27        A        LDB    REGA
D566 E2   E0        A        SBCB   ,S+
D568 20   CB     D535        BRA    SAVEBC

D56A E6   C0        A SBI    LDB    ,U+
D56C 20   F1     D55F        BRA    SBBX

D56E 17   FF14   D485 ANA    LBSR   GETSU
D571 D4   27        A ANA1   ANDB   REGA
D573 1C   DE        A ANA2   ANDCC  #%11011110
D575 20   BE     D535        BRA    SAVEBC

D577 E6   C0        A ANI    LDB    ,U+
D579 20   F6     D571        BRA    ANA1

D57B 17   FF07   D485 XRA    LBSR   GETSU
D57E D8   27        A XRA1   EORB   REGA
D580 1C   DE        A XRA2   ANDCC  #%11011110
D582 20   B1     D535        BRA    SAVEBC

D584 E6   C0        A XRI    LDB    ,U+
```

```
D586 20    F6   D57E          BRA    XRA1

D588 17    FEFA D485 ORAX     LBSR   GETSU
D58B DA    27      A ORAX1    ORB    REGA
D58D 20    F1   D580          BRA    XRA2

D58F E6    C0      A ORI      LDB    ,U+
D591 20    F8   D58B          BRA    ORAX1

D593 17    FEEF D485 CMPX     LBSR   GETSU
D596 34    04      A CMPX1    PSHS   B
D598 D6    27      A          LDB    REGA
D59A E1    E0      A          CMPB   ,S+
D59C 20    99   D537          BRA    SAVECC
D59E E6    C0      A CPI      LDB    ,U+
D5A0 20    F4   D596          BRA    CMPX1

D5A2 E6    5F      A INR      LDB    -1,U
D5A4 54                       LSRB
D5A5 54                       LSRB
D5A6 54                       LSRB
D5A7 C4    07      A          ANDB   #$7
D5A9 C1    06      A          CMPB   #%110
D5AB 26    0B   D5B8          BNE    INR1
D5AD 9E    24      A          LDX    REGH
D5AF 96    26      A          LDA    PSW
D5B1 1F    8A      A          TFR    A,CC
D5B3 6C    84      A          INC    ,X
D5B5 16    FF7F D537          LBRA   SAVECC
D5B8 9E    04      A INR1     LDX    <TOPREG
D5BA 96    26      A          LDA    PSW
D5BC 1F    8A      A          TFR    A,CC
D5BE 6C    85      A          INC    B,X
D5C0 16    FF74 D537          LBRA   SAVECC

D5C3 E6    5F      A DCR      LDB    -1,U
D5C5 54                       LSRB
D5C6 54                       LSRB
D5C7 54                       LSRB
D5C8 C4    07      A          ANDB   #$7
D5CA C1    06      A          CMPB   #%110
D5CC 26    0B   D5D9          BNE    DCR1
D5CE 9E    24      A          LDX    REGH
D5D0 96    26      A          LDA    PSW
D5D2 1F    8A      A          TFR    A,CC
D5D4 6A    84      A          DEC    ,X
D5D6 16    FF5E D537          LBRA   SAVECC
D5D9 9E    04      A DCR1     LDX    <TOPREG
D5DB 96    26      A          LDA    PSW
D5DD 1F    8A      A          TFR    A,CC
D5DF 6A    85      A          DEC    B,X
D5E1 16    FF53 D537          LBRA   SAVECC

D5E4 DC    20      A INXB     LDD    REGB
D5E6 C3    0001    A          ADDD   #1
D5E9 DD    20      A          STD    REGB
D5EB 39                       RTS

D5EC DC    22      A INXD     LDD    REGD
D5EE C3    0001    A          ADDD   #1
D5F1 DD    22      A          STD    REGD
D5F3 39                       RTS

D5F4 DC    24      A INXH     LDD    REGH
D5F6 C3    0001    A          ADDD   #1
D5F9 DD    24      A          STD    REGH
D5FB 39                       RTS

D5FC 31    21      A INXSP    LEAY   1,Y
D5FE 39                       RTS

D5FF DC    20      A DCXB     LDD    REGB
D601 83    0001    A          SUBD   #1
D604 DD    20      A          STD    REGB
D606 39                       RTS

D607 DC    22      A DCXD     LDD    REGD
D609 83    0001    A          SUBD   #1
D60C DD    22      A          STD    REGD
D60E 39                       RTS

D60F DC    24      A DCXH     LDD    REGH
D611 83    0001    A          SUBD   #1
D614 DD    24      A          STD    REGH
D616 39                       RTS

D617 31    3F      A DCXSP    LEAY   -1,Y
D619 39                       RTS

D61A 08    27      A RLC      LSL    REGA
D61C 24    1E   D63C          BCC    CARRY
D61E 0C    27      A          INC    REGA
D620 20    1A   D63C          BRA    CARRY

D622 04    27      A RRC      LSR    REGA
D624 24    16   D63C          BCC    CARRY
D626 D6    27      A          LDB    REGA
D628 CA    80      A          ORB    #$80
D62A D7    27      A          STB    REGA
D62C 20    0E   D63C          BRA    CARRY

D62E 96    26      A RAL      LDA    PSW
D630 1F    8A      A          TFR    A,CC
D632 09    27      A          ROL    REGA
```

```
D634 20    06   D63C          BRA    CARRY

D636 96    26      A RAR      LDA    PSW
D638 1F    8A      A          TFR    A,CC
D63A 06    27      A          ROR    REGA
D63C D6    26      A CARRY    LDB    PSW
D63E C4    FE      A          ANDB   #$FE
D640 24    02   D644          BCC    CARRY2
D642 CA    01      A          ORB    #1       SET CARRY
D644 D7    26      A CARRY2   STB    PSW
D646 39                       RTS

D647 03    27      A CMA      COM    REGA
D649 39                       RTS

D64A 96    27      A DAAX     LDA    REGA
D64C 8B    00      A          ADDA   #0
D64E 19                       DAA
D64F 97    27      A          STA    REGA
D651 16    FEE3 D537          LBRA   SAVECC

D654 D6    26      A STC      LDB    PSW
D656 CA    01      A          ORB    #1
D658 D7    26      A          STB    PSW
D65A 39                       RTS

D65B D6    26      A CMC      LDB    PSW
D65D C8    01      A          EORB   #1
D65F D7    26      A          STB    PSW
D661 39                       RTS

D662 EC    C1      A JUMP     LDD    ,U++
D664 1E    89      A JUMPX    EXG    A,B
D666 1F    03      A          TFR    D,U
D668 39                       RTS

D669 DE    24      A PCHL     LDU    REGH
D66B 39                       RTS

D66C 33    42      A NOJUMP   LEAU   2,U
D66E 39                       RTS

                     JUMPC    MACR
                      LDA PSW
                      BITA #%0
                      BNE JUMP
                      BRA NOJUMP
                      ENDM

                     JUMPM    MACR
                      LDA PSW
                      BITA #%0
                      BEQ JUMP
                      BRA NOJUMP
                      ENDM

D66F                 JC       JUMPC  $1
D677                 JNC      JUMPM  $1
D67F                 JZ       JUMPC  $40
D687                 JNZ      JUMPM  $40
D68F                 JP       JUMPM  $80
D697                 JM       JUMPC  $80
D69F                 JPE      JUMPM  $4
D6A7                 JPO      JUMPC  $4

                     * RESTART OPERATION

D6AF E6    5F      A RST      LDB    -1,U
D6B1 C4    38      A          ANDB   #$38 GET RESTART DATA AAA

                     *  ACC D CONTENTS NEW RESTART ADDRESS

D6B3 4F                       CLRA
D6B4 20    04   D6BA          BRA    CALLZZ

D6B6 EC    C1      A CALL     LDD    ,U++
D6B8 1E    69      A          EXG    A,B
D6BA 1E    03      A CALLZZ   EXG    D,U
D6BC 1E    89      A          EXG    A,B
D6BE ED    A3      A          STD    ,--Y
D6C0 39                       RTS

D6C1 33    42      A NOCALL   LEAU   2,U
D6C3 39                       RTS

                     CALLC    MACR
                      LDA PSW
                      BITA #%0
                      BEQ CALL
                      BRA NOCALL
                      ENDM

                     CALLM    MACR
                      LDA PSW
                      BITA #%0
                      BNE CALL
                      BRA NOCALL
                      ENDM

D6C4                 CCALL    CALLM  $1
D6CC                 CNC      CALLC  $1
D6D4                 CZ       CALLM  $40
D6DC                 CNZ      CALLC  $40
```

```
D4E4                    CP     CALLC  $80
D4EC                    CM     CALLM  $80
D4F4                    CPE    CALLM  $4
D4FC                    CPO    CALLC  $4

D704 EC    A1         A RET    LDD    ,Y++
D706 1E    89         A        EXG    A,B
D708 1F    03         A        TFR    D,U
D70A 39                        RTS

                        RETC   MACR
                         LDA PSW
                         BITA 0#0
                         BEQ RET
                         RTS
                         ENDM

                        RETM   MACR
                         LDA PSW
                         BITA 0#0
                         BNE RET
                         RTS
                         ENDM

D70B                    RC     RETM   $1
D712                    RNC    RETC   $1
D719                    RZ     RETM   $40
D720                    RNZ    RETC   $40
D727                    RP     RETC   $80
D72E                    RM     RETM   $80
D735                    RPE    RETM   $4
D73C                    RPO    RETC   $4

D743 96    0E         A IN     LDA    LOC
D745 E6    C0         A        LDB    ,U+
D747 1F    01         A        TFR    D,X
D749 E6    84         A        LDB    ,X
D74B D7    27         A        STB    REGA
D74D 39                        RTS
```

```
D74E 96    0E         A OUT    LDA    LOC
D750 E6    C0         A        LDB    ,U+
D752 1F    01         A        TFR    D,X
D754 D6    27         A        LDB    REGA
D756 E7    84         A        STB    ,X
D758 39                        RTS

D759 C6    FF         A EI     LDB    #$FF
D75B D7    2C         A        STB    INTE
D75D 39                        RTS
D75E 0F    2C         A DI     CLR    INTE
D760 39                        RTS

                        PUSHX  MACR
                         LDD #0
                         EXG A,B
                         STD ,--Y
                         RTS
                         ENDM

                        POPX   MACR
                         LDD ,Y++
                         EXG A,B
                         STD #0
                         RTS
                         ENDM

D761                    PUSHW  PUSHX  PSW
D768                    PUSHB  PUSHX  REGB
D76F                    PUSHH  PUSHX  REGH
D776                    PUSHD  PUSHX  REGD
D77D                    POPW   POPX   PSW
D784                    POPB   POPX   REGB
D78B                    POPD   POPX   REGD
D792                    POPH   POPX   REGH

D799 39                 NOP    RTS

           D100       A        END    START
```

8080仿真程序/调试程序机器码清单

```
D100  20 03 12 20 2F 1F 50 4A 1F 8B 8E F0 15 9F 00 8E  .. /.PJ........
D110  F0 18 9F 02 8E F0 0F 9F 0A 8E F3 CD 9F 08 C6 20  ...............
D120  DD 04 86 E0 97 0E 9E 04 C6 26 6F 00 5A 26 FB CE  ..........&o.Z&..
D130  00 03 DF 2A 17 01 94 86 3A 8D 43 8D 3E 81 20 10  ...*....:.C.>. .
D140  27 00 C1 97 06 8D 34 81 18 27 E9 97 07 DC 06 30  '......4..'.....0
D150  8D 00 52 34 10 30 8D 00 28 10 A3 84 27 0E 30 04  ..R4.0..(...'.0.
D160  AC E4 26 F5 35 10 86 3F 8D 14 20 C8 32 62 86 20  ..&.5..?.. .2b.
D170  8D 0C EC 02 30 8C 0A AD 8B 20 B9 16 01 1E 16 01  ....0.... ......
D180  27 4E 45 00 A1 50 43 00 8E 52 47 00 EF 47 4F 00  'NE..PC..RG..GO.
D190  B0 45 58 00 EB 42 4B 00 DC 54 52 00 92 4D 45 00  .EX..BK..TR..ME.
D1A0  24 44 4B 00 C7 8D 59 C6 0C 1F B8 1F 01 17 01 1B  $DK...Y.........
D1B0  17 01 10 9E 0C 17 01 0D 17 00 E1 81 0D 26 01 39  ..............&.9
D1C0  81 0A 26 08 9E 0C 30 01 9F 0C 20 DB 81 2F 26 02  ..&...0... ../&.
D1D0  20 D5 81 20 26 06 9E 0C 30 1F 20 EC 80 30 24 01   .. &...0. ..0$.
D1E0  39 81 0A 25 0C 81 11 24 01 39 80 07 81 10 25 01  9..%...$.9....%.
D1F0  39 9E 0C C6 04 68 84 5A 26 FB AB 84 A7 84 20 B8  9....h.Z&...... .
D200  C6 0C 20 5B 8D 1C 16 FF 2E 30 8D 00 C6 20 7B C6  .. [.....0... ..
D210  2A 20 4C 8D 0D 34 10 8D 05 35 10 24 F6 39 9E 08  * L..4...5.$.9..
D220  1F 15 DE 2A 10 9E 28 17 01 E6 DF 2A 10 9F 28 20  ...*..(....*..(
D230  41 17 00 97 DE 2A 10 9E 28 17 01 D4 11 93 2D 26  A....*..(.....-&
D240  F8 DF 2A 10 9F 28 20 28 8E 20 4C BF FF F8 8E F8  ..*..( (. L.....
D250  27 BF FF FA 8E F8 C3 BF FF FC 7E E8 00 C6 2D 1F  '.............-.
D260  B8 1F 01 10 9E 0A 1F 25 8D 9F 8D 06 6E 9F FF FE  .......%....n...
D270  8D 97 9E 04 8D 4D 8D 4B 8D 49 8D 49 8D 47 8D 43  .....M.K.I.I.G.C
D280  8D 41 8D 41 8D 3D 20 43 8D 1E A6 80 81 04 26 F8  .A.A.= C......&.
D290  39 8D 09 1F 89 16 02 9D 9E 00 1F 15 34 10 8D F8  9............4...
D2A0  35 90 9E 02 1F 15 96 27 34 10 8D F6 35 90 A6 84  5......'4...5...
D2B0  C6 10 3D 8D 02 A6 80 84 0F 8B 30 81 3A 25 E9 8B  ..=.......0.:%..
D2C0  07 20 E5 8D E9 8D E7 86 20 20 DD 86 0A 8D D9 86  . ......  ......
D2D0  0D 20 D5 0D 0A 42 20 43 20 20 44 20 45 20 20 48  . ...B C  D E  H
D2E0  20 4C 20 20 57 20 20 41 20 20 53 50 20 20 20 50   L  W  A  SP   P
D2F0  43 20 20 20 49 4E 54 45 20 42 4B 50 54 0D 0A 04  C   INTE BKPT...
D300  07 ED 01 F9 02 2C 02 5C 02 6E 02 7B 02 88 02 93  .......,.¥.n......
D310  04 69 01 59 01 3D 02 D4 02 92 02 B3 01 32 03 0A  ...Y.=.......2..
D320  04 89 01 FB 01 4B 02 EF 02 92 02 B3 01 32 03 12  .....K.......2..
D330  04 89 01 60 01 44 02 DC 02 92 02 B3 01 32 03 1E  ...'.D.......2..
D340  04 89 02 10 01 52 02 F7 02 92 02 B3 01 32 03 26  .....R.......2.&
```

续表

```
D350  04 89 01 67 01 AC 02 E4 02 92 02 B3 01 32 03 3A ...9........2.:
D360  04 89 02 14 01 B5 02 FF 02 92 02 B3 01 32 03 37 .............2.7
D370  FF 81 01 6E 01 9E 02 EC 02 92 02 B3 01 32 03 44 ...n.........2.D
D380  FF 96 02 18 01 A5 03 07 02 92 02 B3 01 32 03 4B .............2.K
D390  03 90 03 F4 02 F7 02 D2 03 4C 03 D8 01 63 03 1F ..........L...c..
D3A0  03 89 03 74 02 EF 02 D2 03 44 03 26 01 77 03 1F ...t.....D.&.w..
D3B0  03 82 03 FB 02 E7 03 BE 03 3C 03 E6 01 C6 03 1F .........<......
D3C0  03 7B 03 74 02 DF 03 B3 03 34 03 26 01 DA 03 1F ...t.....4.&....
D3D0  03 AC 04 02 03 17 01 4E 03 6C 03 DF 01 E7 03 1F ...~...N.l......
D3E0  03 A5 02 D9 03 0F 01 3E 03 64 03 26 01 F4 03 1F ........>.d.&....
D3F0  03 97 03 ED 02 FF 03 CE 03 54 03 D1 01 FF 03 1F ..........T......
D400  03 9E 00 B9 03 07 03 C9 03 5C 03 26 02 0E 03 1F ..........¥.&....
D410  E6 C0 2B 17 C5 40 27 08 C1 76 10 27 FE 4A 20 77 ..+..@'..v.'.J w
D420  30 8D FE EC 4F 58 49 EC 8B 6E 8B C5 40 26 0B 54 0...OXI..n..@&.T
D430  54 54 C4 07 30 8D FE C8 20 EA 30 8D FF 52 C4 3F TT..0... .0..R.?
D440  20 E2 E6 C4 8D 53 33 41 39 10 9E 24 39 9E 20 96  ....S3A9..$9. .
D450  27 A7 84 39 9E 22 96 27 A7 84 39 9E 20 A6 84 97 '..9.".'..9. ...
D460  27 39 9E 22 A6 84 97 27 39 EC C1 1E 89 DD 20 39 '9."...'9..... 9
D470  EC C1 1E 89 DD 22 39 EC C1 1E 89 DD 24 39 EC C1 ....."9.....$9..
D480  1E 89 1F 02 39 A6 5F 84 07 81 06 26 05 9E 24 E6 ....9._.....&..$.
D490  84 39 9E 04 E6 86 39 8D EC A6 5F 44 44 44 84 07 .9....9..._DDD..
D4A0  81 06 26 05 9E 24 E7 84 39 9E 04 E7 86 39 8D 27 ..&..$..9....9.'
D4B0  96 27 A7 84 39 8D 20 E6 84 D7 27 39 8D 19 DC 24 .'..9. ...'9...$
D4C0  1E 89 ED 84 39 8D 10 EC 84 1E 89 DD 24 39 DC 22 ....9........$9."
D4D0  9E 24 9F 22 DD 24 39 EC C1 1E 89 1F 01 39 DC 24 .$.".$9......9.$
D4E0  1E 89 AE A4 ED A4 1F 10 1E 89 DD 24 39 8D 96 DB ............$9...
D4F0  27 20 42 D6 27 EB C0 20 3C 8D 8A DB 27 D7 27 D6 ' B.'.. <...'.'.
D500  26 C4 01 D9 27 20 2E E6 C0 20 F0 DC 20 D3 24 DD &...' ... .. .$.
D510  24 1F A8 84 01 D6 26 C4 FE D7 26 9A 26 97 26 39 $.....&...&.&.&9
D520  DC 22 20 E9 DC 24 20 E5 1F 20 20 E1 17 FF 56 34 ." ..$ ..   ...V4
D530  04 D6 27 E0 E0 D7 27 1F A8 34 02 1F 89 84 01 8A ..'...'..4......
D540  02 97 26 54 C4 10 DA 26 D7 26 35 02 C6 10 3D C4 ..&T...&.&5...=.
D550  C0 DA 26 D7 26 39 D6 27 E0 C0 20 D9 17 FF 26 34 ..&.&9.'.. ...&4
D560  04 96 26 44 D6 27 E2 E0 20 CB E6 C0 20 F1 17 FF ..&D.'.. ... ...
D570  14 D4 27 1C DE 20 BE E6 C0 20 F6 17 FF 07 D8 27 ..'.. ... .....'
D580  1C DE 20 B1 E6 C0 20 F6 17 FE FA DA 27 20 F1 E6 .. ... .....' ..
D590  C0 20 F8 17 FE EF 34 04 D6 27 E1 E0 20 99 E6 C0 . ....4..'.. ...
D5A0  20 F4 E6 5F 54 54 54 C4 07 C1 06 26 0B 9E 24 96  ..._TTT....&..$.
D5B0  26 1F 8A 6C 84 16 FF 7F 9E 04 96 26 1F 8A 6C 85 &..l.......&..l.
D5C0  16 FF 74 E6 5F 54 54 54 C4 07 C1 06 26 0B 9E 24 ..t._TTT....&..$
D5D0  96 26 1F 8A 6A 84 16 FF 5E 9E 04 96 26 1F 8A 6A .&..j...^...&..j
D5E0  85 16 FF 53 DC 20 C3 00 01 DD 20 39 DC 22 C3 00 ...S. .... 9."..
D5F0  01 DD 22 39 DC 24 C3 00 01 DD 24 39 31 21 39 DC .."9.$....$91!9.
D600  20 83 00 01 DD 20 39 DC 22 83 00 01 DD 22 39 DC  .... 9."...."9.
D610  24 83 00 01 DD 24 39 31 3F 39 08 27 24 1E 0C 27 $....$91?9.'$..'
D620  20 1A 04 27 24 16 D6 27 CA 80 D7 27 20 0E 96 26  ..'$..'...' ..&
D630  1F 8A 09 27 20 06 96 26 1F 8A 06 27 D6 26 C4 FE ...' ..&...'.&..
D640  24 02 CA 01 D7 26 39 03 27 39 96 27 8B 00 19 97 $....&9.'9.'....
D650  27 16 FE E3 D6 26 CA 01 D7 26 39 D6 26 C8 01 D7 '....&...&9.&...
D660  26 39 EC C1 1E 89 1F 03 39 DE 24 39 33 42 39 96 &9......9.$93B9.
D670  26 85 01 26 ED 20 F5 96 26 85 01 27 E5 20 ED 96 &..&. ..&..'. ..
D680  26 85 40 26 DD 20 E5 96 26 85 40 27 D5 20 DD 96 &.@&. ..&.@'. ..
D690  26 85 80 27 CD 20 D5 96 26 85 80 26 C5 20 CD 96 &..'. ..&..&. ..
D6A0  26 85 04 27 BD 20 C5 96 26 85 04 26 B5 20 BD E6 &..'. ..&..&. ..
D6B0  5F C4 38 4F 20 04 EC C1 1E 89 1E 03 1E 89 ED A3 _.8O ...........
D6C0  39 33 42 39 96 26 85 01 26 EC 20 F5 96 26 85 01 93B9.&..&. ..&..
D6D0  27 E4 20 ED 96 26 85 40 26 DC 20 E5 96 26 85 40 '. ..&.@&. ..&.@
D6E0  27 D4 20 DD 96 26 85 80 27 CC 20 D5 96 26 85 80 '. ..&..'. ..&..
D6F0  26 C4 20 CD 96 26 85 04 26 BC 20 C5 96 26 85 04 &. ..&..&. ..&..
D700  27 B4 20 BD EC A1 1E 89 1F 03 39 96 26 85 01 26 '. .......9.&..&
D710  F3 39 96 26 85 01 27 EC 39 96 26 85 40 26 E5 39 .9.&..'.9.&.@&.9
D720  96 26 85 40 27 DE 39 96 26 85 80 27 D7 39 96 26 .&.@'.9.&..'.9.&
D730  85 80 26 D0 39 96 26 85 04 26 C9 39 96 26 85 04 ..&.9.&..&.9.&..
D740  27 C2 39 96 0E E6 C0 1F 01 E6 84 D7 27 39 96 0E '.9.........'9..
D750  E6 C0 1F 01 D6 27 E7 84 39 C6 FF D7 2C 39 0F 2C .....'..9...,9.,
D760  39 DC 26 1E 89 ED A3 39 DC 20 1E 89 ED A3 39 DC 9.&....9. ....9.
D770  24 1E 89 ED A3 39 DC 22 1E 89 ED A3 39 EC A1 1E $....9."....9...
D780  89 DD 26 39 EC A1 1E 89 DD 20 39 EC A1 1E 89 DD ..&9..... 9.....
D790  22 39 EC A1 1E 89 DD 24 39 39 00 00 00 00 00 00 "9.....$99......
```

该程序本身（$5009～$502C）的检查和为$001280（不受存储单元位置的影响）。

仿真程序/调试程序的检查和为$027463（$D100～$D799）。

下面介绍往BASIC WASTER-3移植的例子。从$7100地址开始输入原始程序之后，请做下面的修改和补充：

```
地址   新数据
710B   77 CE
7110   77 D5
7115   77 A3
711A   77 F1
71BC   1B
71C1   0D
7248   32 62 4F 1F 8B 7E F2 A9
726D
 :
779A   86 70 1F 2B 7E 71
77A0   34 00 00 8D 07 A7 84 8D 03 A7 01 39 8D 0D 48 48
77B0   48 4B B7 77 A1 8D 04 BA 77 A1 39 8D 11 80 30 25
77C0   FA 81 0A 25 08 80 07 25 F2 81 10 24 EE 39 8D 18
77D0   BD E8 04 20 06 34 02 8D 0F 35 02 BD E8 20 34 02
77E0   B6 77 A2 1F 8B 35 02 39 4F 1E 8B B7 77 A2 0F 9E
77F0   39 34 02 8D D9 81 1B 35 02 27 03 1C FE 39 1A 01
7800   39
```

该结果的检查和为$029603（$7100～$7800）。TR命令和ME命令的操作规定不同，TR命令是每按一次键，执行一步，而ME命令是用RETURN键前进一个地址，用ESC键使命令结束。同时，用EXEC&H779A可以进行软起动。

然而，移植后就不能再定位。同时，DK命令和EX命令是相同的。用GO命令和TR命令作仿真IN命令和OUT命令时的操作是不能进行的。上述移植只是移植调试程序那部分的内容。

5.2.2 8080交叉反汇编程序

该程序是用位置独立技巧的方法，按会话方式在RAM上面进行编写和工作的。所需要的RAM区大约为1.5K字节。执行的起始地址是装入本程序的起始地址。程序清单假定在$8000地址开始（参见表5.7）。

程序起动之后，本程序的标题信息“QUIT Y/N？”将在终端上显示出来，并成为等待是否停止执行的输入。

如果是继续执行时，输入N键。反之，如果要结束，输入Y键，则从MC6809的RESET向量地址处返回到MC6809系统。

其次，因为需要知道反汇编的起始地址和结束地址，所以要输入4位十六进制数，如果结束地址的地址数小，则终端上仍为“QUIT Y/N？”状态。如果输入了正确的地址数据，接着就要知道“EXIT Y/N？”，如数据正确，需要继续执行时，要输入Y键，则反汇编的程

序清单被打印到终端上。十六进制的数据在 $ 符号之后去掉 0 后，显示出十六进制数据，不能输出象在8080中经常使用的 0 0 × × H这种形式。

表5.7　8080交叉反汇编程序清单

执行举例

```
*E
*E 8000;G :9
EXBUG09 2.1
*E MDOS
MDOS09  3.01
=LOAD DASM80.LO:1;V

*E 8000;G
8080 DISASSEMBLER VER 1.0

QUIT Y/N ? N
BEGIN ADDRESS = F200
END   ADDRESS = F210
EXEC Y/N ? Y
0101  F200  A6        ANA    M
0102  F201  84        ADD    H
0103  F202  34        INR   M
0104  F203  02        STAX B
0105  F204  EB        XCHG
0106  F205  E0        RPO
0107  F206  17        RAL
0108  F207  FE  BB    CPI     $BB
0109  F209  7A        MOV   A,D
0110  F20A  FF        RST   07
0111  F20B  91        SUB     C
0112  F20C  26  08    MVI   H,$08
0113  F20E  53        MOV   D,E
0114  F20F  1F        RAR
0115  F210  98        SBB     B

QUIT Y/N ? Y
EXBUG09 2.1
```

8080交叉反汇编程序清单

```
                   NAM    DASM80
                   TTL    8080 CROSS-DISASSEMBLER REV 2.0

                   *
                   * NON SELF-MODIFIED OPERATION CODE SECTION
                   *
0030                      ORG    $8000

        8000     A BASE   EQU    *

8000 16 0220 8223 ENTRY   LBRA   START     NORMAL START
8003 16 8245 824B         LBRA   SOFT      SOFT START

8006    0002     A BEGA   RMB    2         START ADDRESS
8008    0002     A ENDA   RMB    2         STOP ADDRESS

                 * INPUT ADDRESS DATA SUBROUTINE

800A    0002     A .INADD RMB    2

                 * INPUT ONE CHARACTER SUBROUTINE

800C    0002     A .INCH  RMB    2

                 * OUTPUT ONE CHR SUBROUTINE

800E    0002     A .OUTCH RMB    2
8010    0002     A NBLINE RMB    2         LINE COUNT BUFFER
        8012     A LINEBF EQU    *
8012    0004     A LINE   RMB    4         LINE PRINT BUFFER
8016    0002     A        RMB    2         SPACE AREA
8018    0004     A LOC    RMB    4
801C    0002     A        RMB    2
801E    0002     A OPCODE RMB    2
8020    0002     A        RMB    2
8022    0002     A OPLOW  RMB    2
8024    0002     A OPHIGH RMB    2
8026    0002     A        RMB    2
8028    0004     A NEM    RMB    4
802C    0002     A        RMB    2
802E    0001     A REGX   RMB    1
802F    0001     A DELM   RMB    1
8030    0001     A REGY   RMB    1
8031    0004     A MEMRY  RMB    4
8035    0002     A ENDBF  RMB    2

8040                      ORG    BASE+64

        8040     A OPTBL  EQU    *
8040    F901     A        FDB    $F901,$0201,$1201,$0A01
8048    1A01     A        FDB    $1A01,$3203,$3A03,$2203
8050    2A03     A        FDB    $2A03,$EB01,$E301,$C602
8058    CE02     A        FDB    $CE02,$D602,$DE02,$E602
8060    EE02     A        FDB    $EE02,$F602,$FE02,$0701
8068    0F01     A        FDB    $0F01,$1701,$1F01,$2F01
8070    2701     A        FDB    $2701,$3701,$3F01,$C303
8078    E901     A        FDB    $E901,$CD03,$C901,$DB02
8080    D302     A        FDB    $D302,$FB01,$F301,$7601
8088    0001     A        FDB    $0001

        808A     A NUMTBL EQU    *
808A    53       A        FCC    'SPHL  STAX BSTAX
809B    44       A        FCC    'DLDAX BLDAX DSTA
80AE    4C       A        FCC    'LDA   SHLD  LHLD
80BF    20       A        FCC    ' XCHG  XTHL  '
80CC    41       A        FCC    'ADI   ACI   SUI
80DD    20       A        FCC    ' SBIU  ANI   '
80EA    58       A        FCC    'XRI   ORI   CPI
80FC    52       A        FCC    'RLC   RRC   '
8108    52       A        FCC    'RAL   RAR   '
8114    43       A        FCC    'CMA   DAA   STC
8126    43       A        FCC    'CMC   JMP   '
8132    50       A        FCC    'PCHL  CALL  RET
8144    49       A        FCC    'IN    OUT   EI
8156    44       A        FCC    'DI    HLT   NOP

                 * MNEMONIC TABLE

8168    4D       A .MOV   FCC    'MOV'
816B    4D       A .MVI   FCC    'MVI'
816E    4C       A .LXI   FCC    'LXI'
8171    50       A .PSH   FCC    'PSH'
8174    50       A .POP   FCC    'POP'
8177    49       A .INR   FCC    'INR'
817A    44       A .DCR   FCC    'DCR'
817D    49       A .INX   FCC    'INX'
8180    44       A .DCX   FCC    'DCX'
8183    44       A .DAD   FCC    'DAD'
8186    52       A .RST   FCC    'RST'

                 * LXI REGISTER OPTION

8189    42       A LXITB  FCC    'BDHS'
```

续表

```
                       * REG-REG MODIFY

8163        42      A RRTBL  FCC    'BCDEHLMA'

                       * ARITHMETIC OPERATION

8195        41      A ARTBL  FCC    'ADDADCSUBSBB'
81A1        41      A        FCC    'ANAXRAORACMP'

                       * CONDITIONAL JUMP OPERATION

81AD        4A      A JCTBL  FCC    'JNZJZ JNCJC '
81B9        4A      A        FCC    'JPOJPEJP JM '

                       * PUSH/PUL REGISTER ASSIGNMENT

81C5        42      A PPTBL  FCC    'BDHW'

                       * MACRO SECTION

                      EXAM   MACR
                       LDA ,U
                       LEAX ¥0,PCR
                       LDB #¥4
                       EORA #¥1
                       ANDA #¥2
                       LBEQ ¥3
                       ENDM

81C9        0D      A QUIT   FCB    $D,$A
81CB        51      A        FCC    'QUIT Y/N ? '
81D6        04      A        FCB    4
81D7        45      A EXEC   FCC    'EXEC Y/N ? '
81E2        04      A        FCB    4
81E3        42      A MSGBEG FCC    'BEGIN ADDRESS = '
81F3        04      A        FCB    4
81F4        45      A MSGEND FCC    'END   ADDRESS = '
8204        04      A        FCB    4
8205        0D      A TITLE  FCB    $D,$A
8207        38      A        FCC    '8080 DISASSEMBLER
8219        56      A        FCC    'VER 1.0'
8220        0D      A MSGPCR FCB    $D,$A
8222        04      A        FCB    4

                       * HARD START

8223 8E   F00F      A START  LDX    #$F00F
8226 AF   8D FDE0            STX    .INADD,PCR
822A 8E   F015      A        LDX    #$F015
822D AF   8D FDDB            STX    .INCH,PCR
8231 8E   F018      A        LDX    #$F018
8234 AF   8D FDD6            STX    .OUTCH,PCR
8238 8E   0100      A        LDX    #$100
823B AF   8D FDD1            STX    NBLINE,PCR

823F 30   8C C3              LEAX   TITLE,PCR
8242 A6   80        A STRT1  LDA    ,X+
8244 17   01BF 8406          LBSR   OUTCH
8247 81   04        A        CMPA   #4
8249 26   F7   8242          BNE    STRT1

                       * SOFT START

824B 30   8D FF7A   SOFT     LEAX   QUIT,PCR
824F 8D   6D   82BE          BSR    PDATA
8251 8D   54   82A7          BSR    INCHNP
8253 81   59        A        CMPA   #'Y
8255 26   04   825B          BNE    SOFT1
8257 6E   9F FFFE A          JMP    [$FFFE]
825B 8D   40   829D SOFT1    BSR    PCRLF
825D 30   8C 83              LEAX   MSGBEG,PCR
8260 8D   5C   82BE          BSR    PDATA
8262 30   8D FDA0            LEAX   BEGA,PCR
8266 8D   49   82B1          BSR    INADDR
8268 8D   33   829D          BSR    PCRLF
826A 30   8C 87              LEAX   MSGEND,PCR
826D 8D   4F   82BE          BSR    PDATA
826F 30   8D FD95            LEAX   ENDA,PCR
8273 8D   3C   82B1          BSR    INADDR
8275 8D   26   829D          BSR    PCRLF
8277 EC   8D FD8D            LDD    ENDA,PCR
827B A3   8D FD87            SUBD   BEGA,PCR
827F 25   CA   824B          BCS    SOFT
8281 30   8D FF52            LEAX   EXEC,PCR
8285 8D   37   82BE          BSR    PDATA
8287 8D   1E   82A7          BSR    INCHNP
8289 81   59        A        CMPA   #'Y
828B 26   BE   824B          BNE    SOFT

828D EE   8D FD75   CONTNU   LDU    BEGA,PCR
8291 86   0D        A        LDA    #$D
8293 17   0170 8406          LBSR   OUTCH
8296 86   0A        A        LDA    #$A
8298 17   016B 8406          LBSR   OUTCH
829B 20   3E   82DB          BRA    DECODE

829D 34   10        A PCRLF  PSHS   X
829F 30   8D FF7D            LEAX   MSGPCR,PCR
82A3 8D   19   82BE          BSR    PDATA
82A5 35   90        A        PULS   X,PC

                       * INPUT ONE CHARACTER

82A7 34   10        A INCHNP PSHS   X
82A9 30   8D FD5F            LEAX   .INCH,PCR
82AD AD   94        A        JSR    [,X]
82AF 35   90        A        PULS   X,PC

                       * INPUT ADDRESS DATA

82B1 34   20        A INADDR PSHS   Y
82B3 31   8D FD53            LEAY   .INADD,PCR
82B7 AD   B4        A        JSR    [,Y]
82B9 35   A0        A        PULS   Y,PC

                       * PRINT STRINGS

82BB 17   0148 8406 PDATA1   LBSR   OUTCH
82BE A6   80        A PDATA  LDA    ,X+
82C0 81   04        A        CMPA   #4
82C2 26   F7   82BB          BNE    PDATA1
82C4 39                      RTS

                       * CLEAR BUFFER.

82C5 34   16        A CLRBF  PSHS   A,B,X
82C7 86   20        A        LDA    #$20        SPACE
82C9 C6   23        A        LDB    #ENDBF-LINEBF
82CB 30   8D FD43            LEAX   LINEBF,PCR
82CF A7   80        A CLRBF1 STA    ,X+
82D1 5A                      DECB
82D2 26   FB   82CF          BNE    CLRBF1
82D4 CC   0D0A      A        LDD    #$0D0A
82D7 ED   81        A        STD    ,X++
82D9 35   96        A        PULS   A,B,X,PC

          82DB      A DECODE EQU    *

82DB 8D   E8   82C5          BSR    CLRBF

82DD 17   010A 83EA          LBSR   SRCH
82E0 25   16   82F8          BCS    DECODX
82E2 34   16        A        PSHS   X,B,A
82E4 30   8D FDA2            LEAX   NUMTBL,PCR
82E8 C6   06        A        LDB    #6
82EA 3D                      MUL
82EB 30   8B        A        LEAX   D,X
82ED 17   01A5 8495          LBSR   PNEM6
82F0 35   16        A        PULS   A,B,X
82F2 17   013B 8430          LBSR   SAVHEX
82F5 16   011C 8414          LBRA   PLINE

                       * BLOCK EXAMINE

82F8                DECODX   EXAM   .MVI,$6,$C7,MVIRM,2
8308                         EXAM   .MOV,$40,$C0,MOVRR,1
8318                         EXAM   .LXI,$01,$CF,LXIRM,3
8328                         EXAM   ARTBL,$80,$C0,ARITH,1
8338                         EXAM   JCTBL,$C2,$C7,JMPC,3
8348                         EXAM   JCTBL,$C4,$C7,CALCC,3
8358                         EXAM   JCTBL,$C0,$C7,RETCC,1
8368                         EXAM   .PSH,$C5,$CF,PUSHRR,1
8378                         EXAM   .POP,$C1,$CF,POPRR,1
8388                         EXAM   .INR,$04,$C7,INRR,1
8398                         EXAM   .DCR,$05,$C7,DCRR,1
83A8                         EXAM   .INX,$03,$CF,INXR,1
83B8                         EXAM   .DCX,$0B,$CF,DCXR,1
83C8                         EXAM   .DAD,$09,$CF,DADX,1
83D8                         EXAM   .RST,$C7,$C7,RSTN,1
83E8 20   26   8410          BRA    SRCHX

83EA A6   C4        A SRCH   LDA    ,U
83EC C6   25        A        LDB    #37
83EE 30   8D FC4E            LEAX   OPTBL,PCR
83F2 A1   81        A SRCHLP CMPA   ,X++
83F4 27   06   83FC          BEQ    FOUND
83F6 5A                      DECB
83F7 26   F9   83F2          BNE    SRCHLP
83F9 1A   01        A        ORCC   #1
83FB 39                      RTS

83FC 50             FOUND    NEGB
83FD CB   25        A        ADDB   #37
83FF A6   1F        A        LDA    -1,X
8401 1E   89        A        EXG    A,B
8403 1C   FE        A        ANDCC  #$FE
8405 39                      RTS

                       * PRINT OUT ONE CHARACTER FORM ACIA

8406 34   10        A OUTCH  PSHS   X
8408 30   8D FC02            LEAX   .OUTCH,PCR
840C AD   94        A        JSR    [,X]
840E 35   90        A        PULS   X,PC
```

```
                           * UNDEFIND OP CODE

8410 C6    01        A SRCHX  LDB    #1       SET SINGLE BYTE
8412 8D    1C    8430         BSR    SAVHEX

                           * UPDATE PSEDO PC
                           * ACC B CONTENT BYTE LENGTH

8414 33    C5        A PLINE  LEAU   B,U
8416 30    8D FBF8            LEAX   LINEBF,PCR
841A A6    80        A PLINE1 LDA    ,X+
841C 8D    E8    8406         BSR    OUTCH
841E 81    0A        A        CMPA   #$A
8420 26    F8    841A         BNE    PLINE1
8422 11A3 8D FBE1    PLINE2   CMPU   ENDA,PCR
8427 1023 FEB0 82DB           LBLS   DECODE

842B 16    FE1D 824B          LBRA   SOFT

                           * PRINT MNEMONIC AND HEX DATA

842E 8D    54    8484 PNEMHX BSR    PNEM

                           * PRINT HEX DATA ONLY

8430 34    36        A SAVHEX PSHS   A,B,X,Y

                           * SETUP LINE NUMBER

8432 8D    67    849B         BSR    SETNBL
8434 1F    30        A        TFR    U,D
8436 30    8D FBDE            LEAX   LOC,PCR
843A 8D    2E    846A         BSR    OUT2H
843C 1F    98        A        TFR    B,A
843E 8D    2A    846A         BSR    OUT2H
8440 30    8D FBDA            LEAX   OPCODE,PCR
8444 A6    C4        A        LDA    ,U
8446 8D    22    846A         BSR    OUT2H
8448 E6    61        A        LDB    1,S
844A 30    8D FBD4            LEAX   OPLOW,PCR
844E 5A                       DECB
844F 27    0B    845C         BEQ    SAVHXE
8451 A6    41        A        LDA    1,U
8453 8D    15    846A         BSR    OUT2H
8455 5A                       DECB
8456 27    04    845C         BEQ    SAVHXE
8458 A6    42        A        LDA    2,U
845A 8D    0E    846A         BSR    OUT2H
845C E6    61        A SAVHXE LDB    1,S
845E 5A                       DECB
845F 27    07    8468         BEQ    SAVXX
8461 5A                       DECB
8462 1027 0098 84FE           LBEQ   SWAP1
8466 20    7E    84E6         BRA    SWAP2

8468 35    B6        A SAVXX  PULS   A,B,X,Y,PC

846A 34    06        A OUT2H  PSHS   A,B
846C C6    10        A        LDB    #$10
846E 3D                       MUL
846F 8D    08    8479         BSR    OUT2HX
8471 A6    E4        A        LDA    ,S
8473 84    0F        A        ANDA   #$F
8475 8D    02    8479         BSR    OUT2HX
8477 35    86        A        PULS   A,B,PC

8479 8B    30        A OUT2HX ADDA   #$30
847B 81    3A        A        CMPA   #$3A
847D 25    02    8481         BCS    OUT2H1
847F 8B    07        A        ADDA   #$7
8481 A7    80        A OUT2H1 STA    X+
8483 39                       RTS

8484 34    26        A PNEM   PSHS   A,B,Y    PRINT MNEMONIC
8486 C6    03        A        LDB    #3
8488 31    8D FB9C   PNEMX    LEAY   NEM,PCR
848C A6    80        A PNEM1  LDA    ,X+
848E A7    A0        A        STA    ,Y+
8490 5A                       DECB
8491 26    F9    848C         BNE    PNEM1
8493 35    A6        A        PULS   A,B,Y,PC

8495 34    26        A PNEM6  PSHS   A,B,Y
8497 C6    06        A        LDB    #6
8499 20    ED    8488         BRA    PNEMX

                           * SET LINE NUMBER

849B EC    8D FB71   SETNBL   LDD    NBLINE,PCR
849F 1E    98        A        EXG    B,A
84A1 8B    01        A        ADDA   #1
84A3 19                       DAA
84A4 1E    89        A        EXG    A,B
84A6 89    00        A        ADCA   #0
84A8 19                       DAA
84A9 ED    8D FB63            STD    NBLINE,PCR
84AD 30    8D FB61            LEAX   LINE,PCR
84B1 8D    B7    846A         BSR    OUT2H
84B3 1F    98        A        TFR    B,A
84B5 8D    B3    846A         BSR    OUT2H
84B7 39                       RTS

                           * CONDITIONAL JUMP/CALL/RETURN COMMON

84B8 34    36        A CONDT  PSHS   A,B,X,Y
84BA A6    C4        A        LDA    ,U
84BC 84    38        A        ANDA   #$38
84BE C6    60        A        LDB    #$20*3
84C0 3D                       MUL
84C1 30    86        A        LEAX   A,X
84C3 8D    BF    8484         BSR    PNEM
84C5 35    B6        A        PULS   A,B,X,Y,PC

84C7 8D    03    84CC JMPC    BSR    JMPXX
84C9 16    FF48 8414 PLINXX   LBRA   PLINE

84CC 17    FF61 8430 JMPXX    LBSR   SAVHEX
84CF 8D    E7    84B8         BSR    CONDT
84D1 39                       RTS

84D2 8D    F8    84CC CALCC   BSR    JMPXX
84D4 86    43        A        LDA    #'C
84D6 A7    8D FB4E            STA    NEM,PCR
84DA 20    ED    84C9         BRA    PLINXX

84DC 8D    EE    84CC RETCC   BSR    JMPXX
84DE 86    52        A        LDA    #'R
84E0 A7    8D FB44            STA    NEM,PCR
84E4 20    E3    84C9         BRA    PLINXX

84E6 EC    8D FB38   SWAP2    LDD    OPLOW,PCR
84EA ED    8D FB45            STD    MEMRY+2,PCR
84EE EC    8D FB32            LDD    OPHIGH,PCR
84F2 ED    8D FB3B            STD    MEMRY,PCR
84F6 86    24        A SWAP2X LDA    #'$
84F8 A7    8D FB34            STA    REGY,PCR
84FC 35    B6        A        PULS   A,B,X,Y,PC

84FE EC    8D FB20   SWAP1    LDD    OPLOW,PCR
8502 ED    8D FB2B            STD    MEMRY,PCR
8506 20    EE    84F6         BRA    SWAP2X

8508 34    36        A SETDD  PSHS   A,B,X,Y
850A 31    8D FB20            LEAY   REGX,PCR
850E A6    C4        A        LDA    ,U
8510 84    38        A        ANDA   #$38
8512 C6    20        A        LDB    #$20
8514 3D                       MUL
8515 30    8D FC74            LEAX   RRTBL,PCR
8519 A6    86        A        LDA    A,X
851B A7    A0        A        STA    ,Y+
851D 86    2C        A        LDA    #',
851F A7    A4        A        STA    ,Y
8521 35    B6        A        PULS   A,B,X,Y,PC

8523 34    36        A SETSS  PSHS   A,B,X,Y
8525 A6    C4        A        LDA    ,U
8527 84    07        A        ANDA   #$7
8529 30    8D FC60            LEAX   RRTBL,PCR
852D 31    8D FAFF            LEAY   REGY,PCR
8531 A6    86        A        LDA    A,X
8533 A7    A4        A        STA    Y
8535 35    B6        A        PULS   A,B,X,Y,PC

8537 17    FEF4 842E MOVRR    LBSR   PNEMHX
853A 8D    CC    8508         BSR    SETDD
853C 8D    E5    8523         BSR    SETSS
853E 20    89    84C9         BRA    PLINXX

8540 86    2C        A LXIRM  LDA    #
8542 A7    8D FAE9            STA    DELM,PCR

           8546      A DADX   EQU    *
           8546      A INXR   EQU    *
           8546      A DCXR   EQU    *
8546 17    FEE5 842E          LBSR   PNEMHX
8549 8D    31    857C         BSR    SETXI
854B 16    FEC6 8414          LBRA   PLINE

           854E      A POPRR  EQU    *
           854E      A PUSHRR EQU    *
854E 17    FEDD 842E          LBSR   PNEMHX
8551 8D    3E    8591         BSR    SETPP
8553 16    FEBE 8414          LBRA   PLINE

           8556      A DCRR   EQU    *
8556 17    FED5 842E INRR     LBSR   PNEMHX
8559 8D    AD    8508         BSR    SETDD
855B 86    20        A        LDA    #$20
855D A7    8D FACE            STA    DELM,PCR
8561 16    FEB0 8414          LBRA   PLINE

8564 17    FEC7 842E RSTN     LBSR   PNEMHX
8567 34    06        A        PSHS   A,B
8569 A6    C4        A        LDA    ,U
856B 84    38        A        ANDA   #$38
856D C6    20        A        LDB    #$20
```

```
856F 3D                         MUL
8570 30    8D FABA              LEAX   REGX,PCR
8574 17    FEF3 846A            LBSR   OUT2H
8577 35    06      A            PULS   A,B
8579 16    FE98 8414            LBRA   PLINE

857C 34    16      A SETXI      PSHS   A,B,X
857E 30    8D FC07              LEAX   LXITB,PCR
8582 A6    C4      A            LDA    ,U
8584 84    30      A            ANDA   #$30
8586 C6    10      A            LDB    #$10
8588 3D                         MUL
8589 A6    86      A            LDA    A,X
858B A7    8D FA9F              STA    REGX,PCR
858F 35    96      A            PULS   A,B,X,PC

8591 34    16      A SETPP      PSHS   A,B,X
8593 30    8D FC2E              LEAX   PPTBL,PCR
8597 20    06   859F            BRA    SETZZ

8599 34    16      A SETRR      PSHS   A,B,X
859B 30    8D FBEE              LEAX   RRTBL,PCR
859F A6    C4      A SETZZ      LDA    ,U
85A1 84    30      A            ANDA   #$30
```

```
85A3 C6    10      A            LDB    #$10
85A5 3D                         MUL
85A6 A6    86      A            LDA    A,X
85A8 A7    8D FA82              STA    REGX,PCR
85AC 35    96      A            PULS   A,B,X,PC

85AE 17    FE7D 842E MVIRM      LBSR   PNEMHX
85B1 17    FF54 8508            LBSR   SETDD
85B4 16    FE5D 8414            LBRA   PLINE

85B7 34    04      A ARITH      PSHS   B
85B9 A6    C4      A            LDA    ,U
85BB 84    38      A            ANDA   #$38
85BD C6    60      A            LDB    #$20*3
85BF 3D                         MUL
85C0 30    86      A            LEAX   A,X
85C2 35    04      A            PULS   B
85C4 17    FE67 842E            LBSR   PNEMHX
85C7 17    FF59 8523            LBSR   SETSS
85CA 16    FE47 8414            LBRA   PLINE

           8223    A            END    START
```

8080交叉反汇编程序清单

```
8000   16 02 20 16 02 45 00 00 00 00 00 00 00 00 00 00 .. ..E..........
8010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8040   F9 01 02 01 12 01 0A 01 1A 01 32 03 3A 03 22 03 ..........2.:.".
8050   2A 03 EB 01 E3 01 C6 02 CE 02 D6 02 DE 02 E6 02 *...............
8060   EE 02 F6 02 FE 02 07 01 0F 01 17 01 1F 01 2F 01 ............../.
8070   27 01 37 01 3F 01 C3 03 E9 01 CD 03 C9 01 DB 02 '.7.?...........
8080   D3 02 FB 01 F3 01 76 01 00 01 53 50 48 4C 20 20 ......v...SPHL
8090   53 54 41 58 20 42 53 54 41 58 20 44 4C 44 41 58 STAX BSTAX DLDAX
80A0   20 42 4C 44 41 58 20 44 53 54 41 20 20 20 4C 44  BLDAX DSTA   LD
80B0   41 20 20 20 53 48 4C 44 20 20 4C 48 4C 44 20 20 A   SHLD  LHLD
80C0   58 43 48 47 20 20 58 54 48 4C 20 20 41 44 49 20 XCHG  XTHL  ADI
80D0   20 20 41 43 49 20 20 20 53 55 49 20 20 20 53 42   ACI   SUI   SB
80E0   49 55 20 20 41 4E 49 20 20 20 58 52 49 20 20 20 IU  ANI   XRI
80F0   4F 52 49 20 20 20 43 50 49 20 20 20 52 4C 43 20 ORI   CPI   RLC
8100   20 20 52 52 43 20 20 20 52 41 4C 20 20 20 52 41   RRC   RAL   RA
8110   52 20 20 20 43 4D 41 20 20 20 44 41 41 20 20 20 R   CMA   DAA
8120   53 54 43 20 20 20 43 4D 43 20 20 20 4A 4D 50 20 STC   CMC   JMP
8130   20 20 50 43 48 4C 20 20 43 41 4C 4C 20 20 52 45   PCHL  CALL  RE
8140   54 20 20 20 49 4E 20 20 20 20 4F 55 54 20 20 20 T   IN    OUT
8150   45 49 20 20 20 20 44 49 20 20 20 20 48 4C 54 20 EI    DI    HLT
8160   20 20 4E 4F 50 20 20 20 4D 4F 56 4D 56 49 4C 58   NOP   MOVMVILX
8170   49 50 53 48 50 4F 50 49 4E 52 44 43 52 49 4E 58 IPSHPOPINRDCRINX
8180   44 43 58 44 41 44 52 53 54 42 44 48 53 42 43 44 DCXDADRSTBDHSBCD
8190   45 48 4C 4D 41 41 44 44 41 44 43 53 55 42 53 42 EHLMAADDADCSUBSB
81A0   42 41 4E 41 58 52 41 4F 52 41 43 4D 50 4A 4E 5A BANAXRAORACMPJNZ
81B0   4A 5A 20 4A 4E 13 4A 43 20 4A 50 4F 4A 50 45 4A JZ JNCJC JPOJPEJ
81C0   50 20 4A 4D 20 42 44 48 57 0D 0A 51 55 49 54 20 P JM BDHW..QUIT
81D0   59 2F 4E 20 3F 20 04 45 58 45 43 20 59 2F 4E 20 Y/N ? .EXEC Y/N
81E0   3F 20 04 42 45 47 49 4E 20 41 44 44 52 45 53 53 ? .BEGIN ADDRESS
81F0   20 3D 20 04 45 4E 44 20 20 20 41 44 44 52 45 53  = .END   ADDRES
8200   53 20 3D 20 04 0D 0A 38 30 38 30 20 44 49 53 41 S = ...8080 DISA
8210   53 53 45 4D 42 4C 45 52 20 56 45 52 20 31 2E 30 SSEMBLER VER 1.0
8220   0D 0A 04 8E F0 0F AF 8D FD E0 8E F0 15 AF 8D FD ................
8230   DB 8E F0 18 AF 8D FD D6 8E 01 00 AF 8D FD D1 30 ...............0
8240   8C C3 A6 80 17 01 BF 81 04 26 F7 30 8D FF 7A 8D .........&.0..z.
8250   6D 8D 54 81 59 26 04 6E 9F FF FE 8D 40 30 8C 83 m.T.Y&.n....@0..
8260   8D 5C 30 8D FD A0 8D 49 8D 33 30 8C 87 8D 4F 30 .¥0....I.30...O0
8270   8D FD 95 8D 3C 8D 26 EC 8D FD 8D A3 8D FD 87 25 ....<.&........%
8280   CA 30 8D FF 52 8D 37 8D 1E 81 59 26 BE EE 8D FD .0..R.7...Y&....
8290   75 86 0D 17 01 70 86 0A 17 01 6B 20 3E 34 10 30 u....p....k >4.0
82A0   8D FF 7D 8D 19 35 90 34 10 30 8D FD 5F AD 94 35 .....5.4.0.._..5
82B0   90 34 20 31 8D FD 53 AD B4 35 A0 17 01 48 A6 80 .4 1..S..5...H..
82C0   81 04 26 F7 39 34 16 86 20 C6 23 30 8D FD 43 A7 ..&.94.. .#0..C.
82D0   80 5A 26 FB CC 0D 0A ED 81 35 96 8D E8 17 01 0A .Z&......5......
82E0   25 16 34 16 30 8D FD A2 C6 06 3D 30 8B 17 01 A5 %.4.0.....=0....
82F0   35 16 17 01 3B 16 01 1C A6 C4 30 8D FE 6D C6 02 5...;.....0..m..
8300   88 06 84 C7 10 27 02 A6 A6 C4 30 8D FE 5A C6 01 .....'....0..Z..
8310   88 40 84 C0 10 27 02 1F A6 C4 30 8D FE 50 C6 03 .@...'....0..P..
8320   88 01 84 CF 10 27 02 18 A6 C4 30 8D FE 67 C6 01 .....'....0..g..
```

```
8330   88 80 84 C0 10 27 02 7F A6 C4 30 8D FE 6F C6 03 .....'....0..o..
8340   88 C2 84 C7 10 27 01 7F A6 C4 30 8D FE 5F C6 03 .....'....0.._..
8350   88 C4 84 C7 10 27 01 7A A6 C4 30 8D FE 4F C6 01 .....'.z..0..O..
8360   88 C0 84 C7 10 27 01 74 A6 C4 30 8D FE 03 C6 01 .....'.t..0.....
8370   88 C5 84 CF 10 27 01 D6 A6 C4 30 8D FD F6 C6 01 .....'....0.....
8380   88 C1 84 CF 10 27 01 C6 A6 C4 30 8D FD E9 C6 01 .....'....0.....
8390   88 04 84 C7 10 27 01 BE A6 C4 30 8D FD DC C6 01 .....'....0.....
83A0   88 05 84 C7 10 27 01 AE A6 C4 30 8D FD CF C6 01 .....'....0.....
83B0   88 03 84 CF 10 27 01 8E A6 C4 30 8D FD C2 C6 01 .....'....0.....
83C0   88 0B 84 CF 10 27 01 7E A6 C4 30 8D FD B5 C6 01 .....'....0.....
83D0   88 09 84 CF 10 27 01 6E A6 C4 30 8D FD A8 C6 01 .....'.n..0.....
83E0   88 C7 84 C7 10 27 01 7C 20 26 A6 C4 C6 25 30 8D .....'.. &...%0.
83F0   FC 4E A1 81 27 06 5A 26 F9 1A 01 39 50 CB 25 A6 .N..'.Z&...9P.%.
8400   1F 1E 89 1C FE 39 34 10 30 8D FC 02 AD 94 35 90 .....94.0.....5.
8410   C6 01 8D 1C 33 C5 30 8D FB F8 A6 80 8D E3 81 0A ....3.0.........
8420   26 F8 11 A3 8D FB E1 10 23 FE B0 16 FE 1D 8D 54 &.......#......T
8430   34 36 8D 67 1F 30 30 8D FB DE 8D 2E 1F 98 8D 2A 46.9.00........*
8440   30 8D FB DA A6 C4 8D 22 E6 61 30 8D FB D4 5A 27 0......".a0...Z'
8450   0B A6 41 8D 15 5A 27 04 A6 42 8D 0E E6 61 5A 27 ..A..Z'..B...aZ'
8460   07 5A 10 27 00 98 20 7E 35 B6 34 06 C6 10 3D 8D .Z.'.. .5.4...=.
8470   08 A6 E4 84 0F 8D 02 35 86 8B 30 81 3A 25 02 8B .......5..0.:%..
8480   07 A7 80 39 34 26 C6 03 31 8D FB 9C A6 80 A7 A0 ...94&..1.......
8490   5A 26 F9 35 A6 34 26 C6 06 20 ED EC 8D FB 71 1E Z&.5.4&.. ....q.
84A0   98 8B 01 19 1E 89 89 00 19 ED 8D FB 63 30 8D FB ............c0..
84B0   61 8D B7 1F 98 8D B3 39 34 36 A6 C4 84 38 C6 60 a......946...8.'
84C0   3D 30 86 8D BF 35 B6 8D 03 16 FF 48 17 FF 61 8D =0...5.....H..a.
84D0   E7 39 8D F8 86 43 A7 8D FB 4E 20 ED 8D EE 86 52 .9...C...N ....R
84E0   A7 8D FB 44 20 E3 EC 8D FB 38 ED 8D FB 45 EC 8D ...D ....8...E..
84F0   FB 32 ED 8D FB 3B 86 24 A7 8D FB 34 35 B6 EC 8D .2...;.$...45...
8500   FB 20 ED 8D FB 2B 20 EE 34 36 31 8D FB 20 A6 C4 . ...+ .461.. ..
8510   84 38 C6 20 3D 30 8D FC 74 A6 86 A7 A0 86 2C A7 .8. =0..t.....,.
8520   A4 35 B6 34 36 A6 C4 84 07 30 8D FC 60 31 8D FA .5.46....0..'1..
8530   FF A6 86 A7 A4 35 B6 17 FE F4 8D CC 8D E5 20 89 .....5........ .
8540   86 2C A7 8D FA E9 17 FE E5 8D 31 16 FE C6 17 FE .,.........1....
8550   DD 8D 3E 16 FE BE 17 FE D5 8D AD 86 20 A7 8D FA ..>......... ...
8560   CE 16 FE B0 17 FE C7 34 06 A6 C4 84 38 C6 20 3D .......4....8. =
8570   30 8D FA BA 17 FE F3 35 06 16 FE 98 34 16 30 8D 0......5....4.0.
8580   FC 07 A6 C4 84 30 C6 10 3D A6 86 A7 8D FA 9F 35 .....0..=......5
8590   96 34 16 30 8D FC 2E 20 06 34 16 30 8D FB EE A6 .4.0... .4.0....
85A0   C4 84 30 C6 10 3D A6 86 A7 8D FA 82 35 96 17 FE ..0..=......5...
85B0   7D 17 FF 54 16 FE 5D 34 04 A6 C4 84 38 C6 60 3D ...T..]4....8.'=
85C0   30 86 35 04 17 FE 67 17 FF 59 16 FE 47 00 00 00 0.5...g..Y..G...
```

用其它MC6809系统插入本程序时，和仿真程序一样，只要对INADDR（\$8224，5），INCH（\$822B，C），OUTCH（\$8232，3）等三个子程序的地址进行修改就行了，直接页面寄存器完全不要动。检查和为\$0229D8（\$8000～\$85CC）。

在MB-6809中，从\$7000装入程序的起点后，需要进行如下的修改和补充：\$75CD～\$75E8是INADDR；还要从\$75FF处写跳转到INCH的指令；从\$7604写跳转到OUTCH的指令。这样在其它系统中也可以使用。补充的部分将设有再定位功能。

补充和修改后的检查和为\$023E4E（\$7000～\$7608）。其补充和修改部分如下：

```
地址   新数据
7224   75 CE
722B   75 FF
7232   76 04

75CD   00 8D 07
75D0   A7 84 8D 03 A7 01 39 8D 0D 48 48 48 48 B7 75 CD
75E0   8D 04 BA 75 CD 39 8D 17 80 30 25 0D 81 0A 25 08
75F0   80 07 25 05 81 10 24 01 39 86 20 8D 07 20 E7 0F
7600   9E BD E8 04 0F 9E 7E E8 20
```

5.2.3 6800交叉反汇编程序

该程序可以做到一次反汇编达5个数据块的存储器区，它是在6809系统上反汇编6800的机器字的程序。由于页数的关系，只给出机器码的源程序清单。其使用方法是：在执行时，首先要知道起始地址和结束地址，所以要分别输入十六进制数，如果输入了十六进制数以外的字符，则从起始处重新进入。详细情况，请见执行举例。

起始执行的地址是＄7000，使用MOTOROLA公司的EXBUG（6809用）的系统进行工作。为移植到其它系统上时，则需要插入如下的外部子程序：

外部子程序

EBC0	打印机初始设定
EBCC	打印ACCA内容
F000	监控程序进行初始化
F015	从终端向ACCA输入一字符
F018	从ACCA输出一字符到终端

插入地址

EBC0	715F,60
EBCC	7115,6
F015	71A0,1　71D3,4　7D3A,B
F018	7118,9　7CF3,4

除此之外，还需要把＄70FB,＄70FD地址改为＄39。＄70FB～7109段是在纸带穿孔机上输出一个字符的程序，所以要把ACIA放在＄FCF4，5之上。在该程序中的＄71F2～＄7256处，如果把调用子程序改变为适当的输出程序，那末用S命令，T命令就可以输出源文本内容。

在本程序中，在所给的外部程序内容中，没有使用监控程序进行初始化。若受监控程序进行控制时，需要使用Reset开关。

另外，因为该程序是不能再定位的，所以在＄7000～＄7FFF处，需要实际安排RAM存储器。下面给出机器码程序清单，起点处开始的检查和为＄04362D（＄7000～＄7D51）。

执行举例

```
*E 7000;G
MC6800 CROSS DISASSEMBLER VERSION 1.2

BEG 0100 3000
END 0110 3008
NEXT BLOCK EXISTS ? N
OPTION L,S,T ?
      L= LIST ONLY
      S= SOURCE ONLY
```

续表

```
        T= LIST AND SOURCE
 ? S
  ORG      $3000
  STA  A   $00 ,X
  DEC  B
  BEQ      *+$05      $3008
  INX
  BRA      *+$FA      $3000
  RTS

MC6800 CROSS DISASSEMBLER VERSION 1.2

BEG 3000
END 3008 3001
NEXT BLOCK EXISTS ? Y
BEG 3000 3002
END 3001 3008
NEXT BLOCK EXISTS ? N
OPTION L,S,T ?
        L= LIST ONLY
        S= SOURCE ONLY
        T= LIST AND SOURCE
 ? S
  ORG      $3000
  STA  A   $00 ,X
  ORG      $3002
  DEC  B
  BEQ      *+$05      $3008
  INX
  BRA      *+$FA      $3000
  RTS

MC6800 CROSS DISASSEMBLER VERSION 1.2
```

6800交叉反汇编机器码程序清单

```
7000   7E 71 5E 00 00 00 00 00 00 00 00 00 00 00 00 00
7010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7030   00 00 00 00 00 00 4D 43 36 38 30 30 20 43 52 4F
7040   53 53 20 44 49 53 41 53 53 45 4D 42 4C 45 52 20
7050   56 45 52 53 49 4F 4E 20 31 2E 32 04 50 41 47 45
7060   20 04 4F 50 54 49 4F 4E 20 4C 2C 53 2C 54 20 3F
7070   20 20 0D 0A 20 20 20 20 20 20 4C 3D 20 4C 49 53
7080   54 20 4F 4E 4C 59 20 0D 0A 20 20 20 20 20 20 53
7090   3D 20 53 4F 55 52 43 45 20 4F 4E 4C 59 0D 0A 20
70A0   20 20 20 20 20 54 3D 20 4C 49 53 54 20 41 4E 44
70B0   20 53 4F 55 52 43 45 0D 0A 20 3F 20 04 4E 45 58
70C0   54 20 42 4C 4F 43 4B 20 45 58 49 53 54 53 20 3F
70D0   20 04 20 20 4F 52 47 20 20 20 20 20 24 04 20 20
70E0   46 43 42 20 24 04 A6 84 BD 7C DD BD 71 0A A6 84
70F0   BD 7C E1 BD 71 0A 8D 3B 30 01 39 34 04 F6 FC F4
7100   C5 02 27 F9 B7 FC F5 35 04 39 7D 70 35 26 0B 7D
7110   70 03 27 03 7E EB CC 7E F0 18 20 DF A6 84 BD 7C
7120   DD BD 71 0A A6 84 BD 7C E1 BD 71 0A 30 01 20 B6
7130   86 20 8D D6 39 C6 28 4F BD 71 0A 5A 26 F9 39 A6
7140   84 81 04 27 06 8D C3 30 01 20 F4 39 8D F1 86 0A
7150   8D B8 86 0D 8D B4 4F 8D B1 8D AF 8D AD 39 BD EB
7160   C0 7F 70 03 7F 70 35 8E 70 16 C6 1C 4F A7 84 30
7170   01 5A 26 F9 BD 71 4E 8E 70 36 BD 71 4C BD 7C F7
7180   BE 7C D6 BF 70 16 BE 7C D8 BF 70 22 8E 70 16 C6
7190   04 34 04 BF 70 0B BD 71 4E 8E 70 BD BD 71 3F BD
71A0   F0 15 81 4E 27 23 BD 7C F7 BE 70 0B 30 01 30 01
71B0   B6 7C D6 A7 84 B6 7C D7 A7 01 B6 7C D8 A7 0C B6
71C0   7C D9 A7 0D 35 04 5A 26 C8 BD 71 4E 8E 70 62 BD
71D0   71 3F BD F0 15 81 4C 27 0D 81 53 27 12 81 54 10
71E0   26 FF 7B BD 71 F5 7A 70 03 BD 72 A6 7E 71 5E BD
71F0   71 F5 7E 71 5E 86 FF B7 70 35 BE 70 16 BF 70 33
7200   BD 71 4E BD 72 5A BD 71 35 BE 70 33 AE 84 BF 70
7210   2E BE 70 33 AE 0C 27 3B BF 70 30 BE 70 33 30 01
```

7220	30	01	BF	70	33	BE	70	2E	BF	70	05	BD	73	95	BD	74
7230	15	7D	70	32	27	08	BD	72	88	BD	74	0E	20	03	BD	74
7240	0B	B6	70	30	B1	70	07	22	E5	B6	70	31	B1	70	08	24
7250	DD	20	B6	7F	70	35	BD	72	71	39	34	02	86	10	BD	70
7260	FB	86	30	BD	70	FB	86	12	BD	71	0A	BD	71	35	35	02
7270	39	34	02	BD	71	35	86	14	BD	70	FB	86	10	BD	70	FB
7280	86	39	BD	70	FB	35	02	39	BF	70	0F	8E	70	DE	BD	71
7290	3F	BE	70	05	BD	70	E6	BD	71	4E	BE	70	05	30	01	BF
72A0	70	07	BE	70	0F	39	BE	70	16	BF	70	2E	BF	70	05	BE
72B0	70	22	BF	70	30	BE	70	18	BF	70	33	7F	70	13	86	01
72C0	B7	70	14	7F	70	15	86	35	B7	70	04	BD	71	4E	BD	71
72D0	4E	86	2D	BD	71	0A	BD	71	0A	BD	71	0A	BD	71	4E	BD
72E0	71	4E	8E	70	5C	BD	71	3F	B6	70	15	8B	01	19	B7	70
72F0	15	8E	70	15	BD	70	E6	BD	71	4E	BD	71	4E	BD	71	4E
7300	8E	70	13	BD	71	1C	8E	70	05	BD	71	1C	C6	09	BD	71
7310	30	5A	26	FA	8D	7F	20	1D	B6	70	14	84	0F	26	03	BD
7320	71	4E	BD	73	A5	B6	70	30	B1	70	07	22	08	26	2F	B6
7330	70	31	7E	7C	BF	B6	70	14	8B	01	19	B7	70	14	B6	70
7340	13	89	00	19	B7	70	13	7A	70	04	2B	02	20	CA	7E	72
7350	C6	86	0A	7A	70	04	27	05	BD	71	0A	20	F6	39	BE	70
7360	33	AE	84	BF	70	2E	BE	70	33	AE	0C	27	E4	BF	70	30
7370	BE	70	33	30	01	30	01	BF	70	33	BE	70	2E	BF	70	05
7380	B6	70	14	8B	01	19	B7	70	14	B6	70	13	89	00	19	B7
7390	70	13	7E	73	00	8E	70	D2	BD	71	3F	8E	70	05	BD	71
73A0	1C	BD	71	4E	39	BD	74	15	8E	70	13	BD	71	1C	8E	70
73B0	05	BD	71	1C	BE	70	05	BD	70	E6	BD	71	30	7D	70	32
73C0	27	0F	C6	05	BD	71	30	5A	26	FA	BD	72	88	BD	74	0E
73D0	39	BE	70	0B	A6	07	81	31	26	11	BD	71	30	BD	71	30
73E0	BD	71	30	BD	71	30	BD	71	30	20	1C	81	32	26	10	BE
73F0	70	05	30	01	BD	70	E6	BD	71	30	BD	71	30	20	08	BE
7400	70	05	30	01	BD	71	1C	BD	74	0B	39	BD	74	64	BE	70
7410	07	BF	70	05	39	7F	70	32	BE	70	05	A6	84	5F	8E	75
7420	D2	A1	84	27	0C	5C	30	01	8C	76	97	26	F4	7A	70	32
7430	39	F7	70	12	7F	70	11	1C	FE	78	70	12	79	70	11	78
7440	70	12	79	70	11	78	70	12	79	70	11	8E	76	97	BF	70
7450	0B	B6	70	0C	BB	70	12	B7	70	0C	B6	70	0B	B9	70	11
7460	B7	70	0B	39	BE	70	0B	A6	07	80	30	BB	70	06	B7	70
7470	08	86	00	B9	70	05	B7	70	07	BD	71	30	BD	74	E9	BD
7480	71	30	A6	07	81	31	26	09	A6	03	81	20	26	03	7E	75
7490	10	A6	84	81	42	26	12	A6	01	81	49	27	03	7E	75	68
74A0	A6	02	81	54	27	03	7E	75	68	A6	03	81	41	27	2F	81
74B0	42	27	2B	BD	71	30	BD	71	30	BD	71	30	A6	07	81	31
74C0	27	4E	A6	04	81	23	26	03	7E	75	14	81	58	26	03	7E
74D0	75	3C	A6	07	81	33	27	03	7E	74	FF	7E	75	57	BD	71
74E0	0A	BD	71	30	BD	71	30	20	D3	BD	71	30	A6	84	BD	71
74F0	0A	A6	01	BD	71	0A	A6	02	BD	71	0A	BD	71	30	39	86
7500	24	BD	71	0A	BE	70	05	30	01	BD	70	E6	BD	71	4E	39
7510	BD	71	4E	39	86	23	BD	71	0A	86	24	BD	71	0A	A6	07
7520	81	33	27	0C	BE	70	05	30	01	BD	70	E6	BD	71	4E	39
7530	BE	70	05	30	01	BD	71	1C	BD	71	4E	39	86	24	BD	71
7540	0A	BE	70	05	30	01	BD	70	E6	86	2C	BD	71	0A	86	58
7550	BD	71	0A	BD	71	4E	39	86	24	BD	71	0A	BE	70	05	30
7560	01	BD	71	1C	BD	71	4E	39	BD	71	30	BD	71	30	BD	71
7570	30	86	2A	BD	71	0A	86	2B	BD	71	0A	86	24	BD	71	0A
7580	BE	70	05	A6	01	4C	4C	34	02	B7	70	09	8E	70	09	BD
7590	70	E6	BD	71	30	BD	71	30	BD	71	30	BD	71	30	35	02
75A0	B7	70	0A	2B	05	7F	70	09	20	05	86	FF	B7	70	09	1C
75B0	FE	B6	70	0A	BB	70	06	B7	70	0A	B6	70	09	B9	70	05
75C0	B7	70	09	86	24	BD	71	0A	8E	70	09	BD	71	1C	BD	71
75D0	4E	39	1B	89	99	B9	A9	C9	D9	F9	E9	8B	9B	BB	AB	CB
75E0	DB	FB	EB	84	94	B4	A4	C4	D4	F4	E4	48	58	78	68	47
75F0	57	77	67	24	25	27	2C	2E	22	85	95	B5	A5	C5	D5	F5
7600	E5	2F	23	2D	28	26	2A	20	8D	28	29	11	0C	0E	4F	5F
7610	7F	6F	0A	81	91	B1	A1	C1	D1	F1	E1	43	53	73	63	8C
7620	9C	BC	AC	19	4A	5A	7A	6A	34	09	88	98	B8	A8	C8	D8
7630	F8	E8	4C	5C	7C	6C	31	08	7E	6E	BD	AD	86	96	B6	A6
7640	C6	D6	F6	E6	8E	9E	BE	AE	CE	DE	FE	EE	44	54	74	64
7650	40	50	70	60	01	8A	9A	BA	AA	CA	DA	FA	EA	36	37	32
7660	33	49	59	79	69	46	56	76	66	3B	39	10	82	92	B2	A2
7670	C2	D2	F2	E2	0D	0F	0B	97	B7	A7	D7	F7	E7	9F	BF	AF
7680	DF	FF	EF	80	90	B0	A0	C0	D0	F0	E0	3F	16	06	17	07
7690	4D	5D	7D	6D	30	35	3E	41	42	41	20	20	30	32	31	41
76A0	44	43	41	23	30	32	32	41	44	43	41	24	30	33	32	41

续表

76B0	44	43	41	24	30	34	33	41	44	43	41	58	30	35	32	41
76C0	44	43	42	23	30	32	32	41	44	43	42	24	30	33	32	41
76D0	44	43	42	24	30	34	33	41	44	43	42	58	30	35	32	41
76E0	44	44	41	23	30	32	32	41	44	44	41	24	30	33	32	41
76F0	44	44	41	24	30	34	33	41	44	44	41	58	30	35	32	41
7700	44	44	42	23	30	32	32	41	44	44	42	24	30	33	32	41
7710	44	44	42	24	30	34	33	41	44	44	42	58	30	35	32	41
7720	4E	44	41	23	30	32	32	41	4E	44	41	24	30	33	32	41
7730	4E	44	41	24	30	34	33	41	4E	44	41	58	30	35	32	41
7740	4E	44	42	23	30	32	32	41	4E	44	42	24	30	33	32	41
7750	4E	44	42	24	30	34	33	41	4E	44	42	58	30	35	32	41
7760	53	4C	41	20	30	32	31	41	53	4C	42	20	30	32	31	41
7770	53	4C	20	24	30	36	33	41	53	4C	20	58	30	37	32	41
7780	53	52	41	20	30	32	31	41	53	52	42	20	30	32	31	41
7790	53	52	20	24	30	36	33	41	53	52	20	58	30	37	32	42
77A0	43	43	20	24	30	34	32	42	43	53	20	24	30	34	32	42
77B0	45	51	20	24	30	34	32	42	47	45	20	24	30	34	32	42
77C0	47	54	20	24	30	34	32	42	48	49	20	24	30	34	32	42
77D0	49	54	41	23	30	32	32	42	49	54	41	24	30	33	32	42
77E0	49	54	41	24	30	34	33	42	49	54	41	58	30	35	32	42
77F0	49	54	42	23	30	32	32	42	49	54	42	24	30	33	32	42
7800	49	54	42	24	30	34	33	42	49	54	42	58	30	35	32	42
7810	4C	45	20	24	30	34	32	42	4C	53	20	24	30	34	32	42
7820	4C	54	20	24	30	34	32	42	4D	49	20	24	30	34	32	42
7830	4E	45	20	24	30	34	32	42	50	4C	20	24	30	34	32	42
7840	52	41	20	24	30	34	32	42	53	52	20	24	30	38	32	42
7850	56	43	20	24	30	34	32	42	56	53	20	24	30	34	32	43
7860	42	41	20	20	30	32	31	43	4C	43	20	20	30	32	31	43
7870	4C	49	20	20	30	32	31	43	4C	52	41	20	30	32	31	43
7880	4C	52	42	20	30	32	31	43	4C	52	20	24	30	36	33	43
7890	4C	52	20	58	30	37	32	43	4C	56	20	20	30	32	31	43
78A0	4D	50	41	23	30	32	32	43	4D	50	41	24	30	33	32	43
78B0	4D	50	41	24	30	34	33	43	4D	50	41	58	30	35	32	43
78C0	4D	50	42	23	30	32	32	43	4D	50	42	24	30	33	32	43
78D0	4D	50	42	24	30	34	33	43	4D	50	42	58	30	35	32	43
78E0	4F	4D	41	20	30	32	31	43	4F	4D	42	20	30	32	31	43
78F0	4F	4D	20	24	30	36	33	43	4F	4D	20	58	30	37	32	43
7900	50	58	20	23	30	33	33	43	50	58	20	24	30	34	32	43
7910	50	58	20	24	30	35	33	43	50	58	20	58	30	36	32	44
7920	41	41	20	20	30	32	31	44	45	43	41	20	30	32	31	44
7930	45	43	42	20	30	32	31	44	45	43	20	24	30	36	33	44
7940	45	43	20	58	30	37	32	44	45	53	20	20	30	34	31	44
7950	45	58	20	20	30	34	31	45	4F	52	41	23	30	32	32	45
7960	4F	52	41	24	30	33	32	45	4F	52	41	24	30	34	33	45
7970	4F	52	41	58	30	35	32	45	4F	52	42	23	30	32	32	45
7980	4F	52	42	24	30	33	32	45	4F	52	42	24	30	34	33	45
7990	4F	52	42	58	30	35	32	49	4E	43	41	20	30	32	31	49
79A0	4E	43	42	20	30	32	31	49	4E	43	20	24	30	36	33	49
79B0	4E	43	20	58	30	37	32	49	4E	53	20	20	30	34	31	49
79C0	4E	58	20	20	30	34	31	4A	4D	50	20	24	30	33	33	4A
79D0	4D	50	20	58	30	34	32	4A	53	52	20	24	30	39	33	4A
79E0	53	52	20	58	30	38	32	4C	44	41	41	23	30	32	32	4C
79F0	44	41	41	24	30	33	32	4C	44	41	41	24	30	34	33	4C
7A00	44	41	41	58	30	35	32	4C	44	41	42	23	30	32	32	4C
7A10	44	41	42	24	30	33	32	4C	44	41	42	24	30	34	33	4C
7A20	44	41	42	58	30	35	32	4C	44	53	20	23	30	33	33	4C
7A30	44	53	20	24	30	34	32	4C	44	53	20	24	30	35	33	4C
7A40	44	53	20	58	30	36	32	4C	44	58	20	23	30	33	33	4C
7A50	44	58	20	24	30	34	32	4C	44	58	20	24	30	35	33	4C
7A60	44	58	20	58	30	36	32	4C	53	52	41	20	30	32	31	4C
7A70	53	52	42	20	30	32	31	4C	53	52	20	24	30	36	33	4C
7A80	53	52	20	58	30	37	32	4E	45	47	41	20	30	32	31	4E
7A90	45	47	42	20	30	32	31	4E	45	47	20	24	30	36	33	4E
7AA0	45	47	20	58	30	37	32	4E	4F	50	20	20	30	32	31	4F
7AB0	52	41	41	23	30	32	32	4F	52	41	41	24	30	33	32	4F
7AC0	52	41	41	24	30	34	33	4F	52	41	41	58	30	35	32	4F
7AD0	52	41	42	23	30	32	32	4F	52	41	42	24	30	33	32	4F
7AE0	52	41	42	24	30	34	33	4F	52	41	42	58	30	35	32	50
7AF0	53	48	41	20	30	34	31	50	53	48	42	20	30	34	31	50
7B00	55	4C	41	20	30	34	31	50	55	4C	42	20	30	34	31	52
7B10	4F	4C	41	20	30	32	31	52	4F	4C	42	20	30	32	31	52
7B20	4F	4C	20	24	30	36	33	52	4F	4C	20	58	30	37	32	52
7B30	4F	52	41	20	30	32	31	52	4F	52	42	20	30	32	31	52

```
7B40  4F 52 20 24 30 36 33 52 4F 52 20 58 30 37 32 52
7B50  54 49 20 20 31 30 31 52 54 53 20 20 30 35 31 53
7B60  42 41 20 20 30 32 31 53 42 43 41 23 30 32 32 53
7B70  42 43 41 24 30 33 32 53 42 43 41 24 30 34 33 53
7B80  42 43 41 58 30 35 32 53 42 43 42 23 30 32 32 53
7B90  42 43 42 24 30 33 32 53 42 43 42 24 30 34 33 53
7BA0  42 43 42 58 30 35 32 53 45 43 20 20 30 32 31 53
7BB0  45 49 20 20 30 32 31 53 45 56 20 20 30 32 31 53
7BC0  54 41 41 24 30 34 32 53 54 41 41 24 30 35 33 53
7BD0  54 41 41 58 30 36 32 53 54 41 42 24 30 34 32 53
7BE0  54 41 42 24 30 35 33 53 54 41 42 58 30 36 32 53
7BF0  54 53 20 24 30 35 32 53 54 53 20 24 30 36 33 53
7C00  54 53 20 58 30 37 32 53 54 58 20 24 30 35 32 53
7C10  54 58 20 24 30 36 33 53 54 58 20 58 30 37 32 53
7C20  55 42 41 23 30 32 32 53 55 42 41 24 30 33 32 53
7C30  55 42 41 24 30 34 33 53 55 42 41 58 30 35 32 53
7C40  55 42 42 23 30 32 32 53 55 42 42 24 30 33 32 53
7C50  55 42 42 24 30 34 33 53 55 42 42 58 30 35 32 53
7C60  57 49 20 20 31 32 32 54 41 42 20 20 30 32 31 54
7C70  41 50 20 20 30 32 31 54 42 41 20 20 30 32 31 54
7C80  50 41 20 20 30 32 31 54 53 54 41 20 30 32 31 54
7C90  53 54 42 20 30 32 31 54 53 54 20 24 30 36 33 54
7CA0  53 54 20 58 30 37 32 54 53 58 20 20 30 34 31 54
7CB0  58 53 20 20 30 34 31 57 41 49 20 20 30 39 31 B1
7CC0  70 08 25 03 7E 73 35 7E 73 5E 20 42 45 47 3D 04
7CD0  20 45 4E 44 3D 04 00 00 00 00 00 00 00 44 44 44
7CE0  44 84 0F 8B 30 81 39 2F 02 8B 07 39 A6 80 81 04
7CF0  27 F9 BD F0 18 20 F5 8E 7C CA 8D F0 8D 12 25 F7
7D00  BF 7C D6 8E 7C D0 8D E4 8D 06 25 F7 BF 7C D8 39
7D10  8D 12 25 0F B7 7C DA 8D 0B 25 08 B7 7C DB BE 7C
7D20  DA 1C FE 39 8D 13 25 10 48 48 48 48 B7 7C DC 8D
7D30  08 25 05 BA 7C DC 1C FE 39 BD F0 15 80 30 25 0F
7D40  81 0A 25 08 80 07 25 07 81 10 24 03 1C FE 39 1A
7D50  01 39
```

当向MB-6809移植时，需要进行如下的修改和补充：原来输出到打印机上的内容，现在要全部输出到显示器屏幕上，为此，希望输出多少行，需要输入数字之后才会继续显示。修改补充后的检查和是$\$$043F0A（$\$$7000～$\$$7D68）。其修改和补充部分如下：

```
地址   新数据
70FB   39
70FD   39
7115   7D 57
7118   7D 57
715F   70 FA
71A0   7D 52
71D3   7D 52
72C7   13
734F   7D 64
7353   BD 71 4E 7E 7D 5C
7CF3   7D 57
7D3A   7D 52

7D52   0F 9E BD E8 04 0F 9E 7E E8 20 8D F4 86 13
7D60   B7 70 04 39 8D F6 7E 72 C6
```

5.2.4 6809反汇编程序

该程序是两遍扫描的反汇编程序。执行第一遍（P命令）扫描后，编好了分支转移和数据访问的地址的符号表，给出表格清单。在使用MDOS系统调用时（SCALL），用SCALL宏

字符命令进行反汇编过程。

直接寻址方式时，符号都看作直接页面寄存器是 0 的情况来使用。所以，如果把直接页面寄存器给定 0 以外的数值时，操作数就没给出正确的符号，这点应该注意。

$6003为软起动（不清除符号表）。

执行例是一个把简易求和检查程序反汇编的例子。

当把数据或工作区也作为程序进行反汇编时，其结果就是把未定义的指令来进行反汇编，将会得到不正确的结果，这点也应注意。

该程序由于篇幅所限，也只给出源码清单（表5.8）。

执行的起始地址为$6000，和6800交叉反汇编程序一样，因为要访问外部子程序，移植时，要插入下述地址：

缺省	插入地址	内 容
$F000	$67DF, E0	监控程序的进入
$F015	$656C, D	一字节输入
$F018	$657A, B	一字节输出
$8000	$70E9, A	标题表起点

该程序也不是可再定位的，需要在$6000～$73FF地址装好RAM区。程序的检查和为$0630E7（$6000～$7179）。

表5.8 6809自反汇编程序清单

执行举例

```
MC6809 DISASSEMBLER REV 1.1
COPYRIGHT BY KASE  FEB 80


Z    CLEAR SYMBOL TABLE
A    APPEND SYMBOL
E    EXCHANGE SYMBOL
T    PRINT SYMBOL TABLE
U    DISPLAY USED SYMBOL TABLE AREA
P    EXECUTE SYMBOL TABLE SEARCH / PASS 1
R    RESTORE BEG/END ADDRESS
S    SOURCE CODE OUTPUT
L    LIST ALL CODE
Q    QUIT
H    HELP
C    ENABLE SCALL MACRO

:  R

BEG = 5009
END = 502C
OFFSET = 0000
:  A

$5002  BEGIN
$5004  END+1
$5006  CHKSUM
$
:  T

BEGIN  5002    END+1  5004    CHKSUM 5006
```

```
: P

BEGIN  5002     END+1  5004     CHKSUM 5006     .00000 0000
.00010 5015

: L


0001 5009 33 8C FA       5006             LEAU   CHKSUM,PCR
0002 500C 6F C4                           CLR    ,U
0003 500E 6F 41                           CLR    $01,U
0004 5010 6F 42                           CLR    $02,U
0005 5012 AE 8C ED       5002             LDX    BEGIN,PCR
0006 5015 A6 80                 .00010    LDA    ,X+
0007 5017 AB 42                           ADDA   $02,U
0008 5019 A7 42                           STA    $02,U
0009 501B A6 41                           LDA    $01,U
0010 501D 89 00                           ADCA   #.00000
0011 501F A7 41                           STA    $01,U
0012 5021 A6 C4                           LDA    ,U
0013 5023 89 00                           ADCA   #.00000
0014 5025 A7 C4                           STA    ,U
0015 5027 AC 8C DA       5004             CMPX   END+1,PCR
0016 502A 26 E9          5015             BNE    .00010
0017 502C 39                              RTS

: Q
```

6809自反汇编程序清单

```
6000  7E 67 8C 7E 67 8F 4D 43 36 38 30 39 20 44 49 53
6010  41 53 53 45 4D 42 4C 45 52 20 52 45 56 20 31 2E
6020  31 0D 0A 43 4F 50 59 52 49 47 48 54 20 42 59 20
6030  4B 41 53 45 20 20 46 45 42 20 38 30 0D 0A 0A 0A
6040  5A 20 20 20 43 4C 45 41 52 20 53 59 4D 42 4F 4C
6050  20 54 41 42 4C 45 0D 0A 41 20 20 20 41 50 50 45
6060  4E 44 20 53 59 4D 42 4F 4C 0D 0A 45 20 20 20 45
6070  58 43 48 41 4E 47 45 20 53 59 4D 42 4F 4C 0D 0A
6080  54 20 20 20 50 52 49 4E 54 20 53 59 4D 42 4F 4C
6090  20 54 41 42 4C 45 0D 0A 55 20 20 20 44 49 53 50
60A0  4C 41 59 20 55 53 45 44 20 53 59 4D 42 4F 4C 20
60B0  54 41 42 4C 45 20 41 52 45 41 0D 0A 50 20 20 20
60C0  45 58 45 43 55 54 45 20 53 59 4D 42 4F 4C 20 54
60D0  41 42 4C 45 20 53 45 41 52 43 48 20 2F 20 50 41
60E0  53 53 20 31 0D 0A 52 20 20 20 52 45 53 54 4F 52
60F0  45 20 42 45 47 2F 45 4E 44 20 41 44 44 52 45 53
6100  53 0D 0A 53 20 20 20 53 4F 55 52 43 45 20 43 4F
6110  44 45 20 4F 55 54 50 55 54 0D 0A 4C 20 20 20 4C
6120  49 53 54 20 41 4C 4C 20 43 4F 44 45 0D 0A 51 20
6130  20 20 51 55 49 54 0D 0A 48 20 20 20 48 45 4C 50
6140  0D 0A 43 20 20 20 45 4E 41 42 4C 45 20 53 43 41
6150  4C 4C 20 4D 41 43 52 4F 0D 0A 04 42 45 47 20 3D
6160  20 04 45 4E 44 20 3D 20 04 4F 46 46 53 45 54 20
6170  3D 20 04 43 41 4C 4C 2E 52 45 53 52 56 2E 52 45
6180  4C 45 53 2E 4F 50 45 4E 20 2E 43 4C 4F 53 45 2E
6190  47 45 54 52 43 2E 50 55 54 52 43 2E 52 45 57 4E
61A0  44 2E 47 45 54 4C 53 2E 50 55 54 4C 53 2E 4B 45
61B0  59 49 4E 2E 44 53 50 4C 59 2E 44 53 50 4C 58 2E
61C0  44 53 50 4C 5A 2E 43 4B 42 52 4B 2E 44 52 45 41
61D0  44 2E 44 57 52 49 54 2E 4D 4F 56 45 20 2E 43 4D
61E0  50 41 52 2E 53 54 43 48 42 2E 53 54 43 48 52 2E
61F0  41 4C 50 48 41 2E 4E 55 4D 44 20 2E 41 44 44 41
6200  4D 2E 53 55 42 41 4D 2E 4D 4D 41 20 20 2E 44 4D
6210  41 20 20 2E 4D 44 45 4E 54 2E 4C 4F 41 44 20 2E
6220  44 49 52 53 4D 2E 50 46 4E 41 4D 2E 41 4C 55 53
```

6230	4D	2E	43	48	41	4E	47	2E	4D	44	45	52	52	2E	41	4C
6240	4C	4F	43	2E	44	45	41	4C	43	2E	45	57	4F	52	44	2E
6250	54	58	42	41	20	2E	54	42	41	58	20	2E	58	42	41	58
6260	20	2E	41	44	42	58	20	2E	41	44	41	58	20	2E	41	44
6270	42	41	58	2E	41	44	58	42	41	2E	53	55	42	58	20	2E
6280	53	55	41	58	20	2E	53	55	42	41	58	2E	53	55	58	42
6290	41	2E	43	50	42	41	58	2E	41	53	52	58	20	2E	41	53
62A0	4C	58	20	2E	50	53	48	58	20	2E	50	55	4C	58	20	2E
62B0	50	52	49	4E	54	2E	50	52	49	4E	58	2E	47	45	54	46
62C0	44	2E	50	55	54	46	44	2E	50	55	54	45	46	2E	45	52
62D0	45	41	44	2E	45	57	52	49	54	2E	4D	52	45	41	44	2E
62E0	4D	57	52	49	54	2E	4D	45	52	45	44	2E	4D	45	57	52
62F0	54	2E	42	4F	4F	54	20	2C	52	2B	20	2C	52	2B	2B	2C
6300	2D	52	20	2C	2D	2D	52	2C	52	20	20	42	2C	52	20	41
6310	2C	52	20	20	20	20	20	24	4E	4E	2C	52	20	20	20	24
6320	4E	4E	4E	4E	2C	52	20	20	20	20	20	20	20	20	20	44
6330	2C	52	20	20	20	20	20	24	4E	4E	20	20	20	20	20	24
6340	4E	4E	4E	4E	20	20	20	20	20	20	20	20	20	20	20	24
6350	4E	4E	4E	4E	20	20	20	44	20	58	20	59	20	55	20	53
6360	20	50	43	20	20	20	20	41	20	42	20	43	43	44	50	20
6370	20	20	20	20	20	20	20	42	53	52	20	53	57	49	20	53
6380	57	49	33	43	4D	50	55	43	4D	50	53	43	4D	50	44	43
6390	4D	50	59	4C	44	59	20	53	54	59	20	4C	44	53	20	53
63A0	54	53	20	4E	45	47	20	20	20	20	20	20	20	20	20	43
63B0	4F	4D	20	4C	53	52	20	20	20	20	20	52	4F	52	20	41
63C0	53	52	20	41	53	4C	20	52	4F	4C	20	44	45	43	20	20
63D0	20	20	20	49	4E	43	20	54	53	54	20	4A	4D	50	20	43
63E0	4C	52	20	20	20	20	20	20	20	20	20	4E	4F	50	20	53
63F0	59	4E	43	20	20	20	20	20	20	20	20	4C	42	52	41	4C
6400	42	53	52	20	20	20	20	44	41	41	20	4F	52	43	43	20
6410	20	20	20	41	4E	44	43	53	45	58	20	45	58	47	20	54
6420	46	52	20	42	52	41	20	42	52	4E	20	42	48	49	20	42
6430	4C	53	20	42	43	43	20	42	43	53	20	42	4E	45	20	42
6440	45	51	20	42	56	43	20	42	56	53	20	42	50	4C	20	42
6450	4D	49	20	42	47	45	20	42	4C	54	20	42	47	45	20	42
6460	4C	45	20	4C	45	41	58	4C	45	41	59	4C	45	41	53	4C
6470	45	41	55	50	53	48	53	50	55	4C	53	50	53	48	55	50
6480	55	4C	55	20	20	20	20	52	54	53	20	41	42	58	20	52
6490	54	49	20	43	57	41	49	4D	55	4C	20	20	20	20	20	53
64A0	57	49	20	53	55	42	41	43	4D	50	41	53	42	43	41	53
64B0	55	42	44	41	4E	44	41	42	49	54	41	4C	44	41	20	53
64C0	54	41	20	45	4F	52	41	41	44	43	41	4F	52	41	20	41
64D0	44	44	41	43	4D	50	58	4A	53	52	20	4C	44	58	20	53
64E0	54	58	20	53	55	42	42	43	4D	50	42	53	42	43	42	41
64F0	44	44	44	41	4E	44	42	42	49	54	42	4C	44	42	20	53
6500	54	42	20	45	4F	52	42	41	44	43	42	4F	52	42	20	41
6510	44	44	42	4C	44	44	20	53	54	44	20	4C	44	55	20	53
6520	54	55	20	00	00	00	00	00	00	00	00	00	00	00	00	00
6530	20	20	20	20	20	20	20	20	00	00	00	00	00	00	00	00
6540	00	00	00	00	3C	00	00	00	00	00	00	00	00	00	00	00
6550	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6560	00	00	00	00	00	00	00	00	00	00	00	7E	F0	15	81	0D
6570	27	0A	81	20	25	03	7C	65	6A	7E	F0	18	7F	65	6A	20
6580	F8	86	0D	8D	E9	86	0A	20	E5	8D	F6	20	02	8D	DF	A6
6590	84	30	01	81	04	26	F6	39	86	20	8D	D2	86	20	20	CE
65A0	8D	F6	8D	F4	20	F6	8D	08	8D	06	20	F0	8D	02	20	F2
65B0	A6	84	44	44	44	44	8D	1B	A6	84	8D	17	30	01	39	34
65C0	02	F1	65	6A	27	0A	B6	20	BD	65	6E	F1	65	6A	26	F6
65D0	35	02	39	8D	03	7E	65	6E	84	0F	8B	30	81	3A	25	02
65E0	8B	07	39	BD	65	6B	80	30	25	0F	81	0A	25	08	80	07
65F0	25	07	81	10	24	03	1C	FE	39	1A	01	39	BD	65	E3	25
6600	11	48	48	48	48	B7	65	38	BD	65	E3	25	05	BA	65	38
6610	1C	FE	39	BD	E7	25	0F	B7	65	3A	8D	E0	25	08	B7	65
6620	3B	BE	65	3A	1C	FE	39	BF	65	3A	B6	65	3B	BB	65	53
6630	B7	65	3B	B6	65	3A	B9	65	52	B7	65	3A	39	7D	65	58
6640	26	41	8E	65	23	BD	65	A6	BE	65	25	8D	DA	8E	65	3A
6650	BD	65	A6	BE	65	25	A6	84	81	10	27	04	81	11	26	03
6660	BD	65	B0	BD	65	B0	8D	71	05	12	8D	65	BF	7D	65	47
6670	27	0E	BE	65	2C	BD	66	27	8E	65	3A	BD	65	A6	20	03
6680	BD	65	A0	BE	65	25	BD	66	27	BE	65	3A	BF	65	61	BD
6690	6F	AB	25	11	BE	65	5B	A6	84	BD	65	6E	30	01	8C	65

续表

```
66A0  61 26 F4 20 06 BD 65 A0 BD 65 9C BD 65 A2 BE 65
66B0  2E 7D 65 68 27 09 86 4C BD 65 6E C6 03 20 02 C6
66C0  04 A6 84 BD 65 6E 30 01 5A 26 F6 BD 65 98 7D 65
66D0  48 27 05 86 5B BD 65 6E 7D 65 47 26 60 7D 65 55
66E0  27 03 7E 67 68 7D 65 56 27 03 7E 67 84 8E 65 30
66F0  BD 71 42 20 07 65 6E 30 01 5A 26 F6 7D 65 67 26
6700  05 7D 65 66 27 0D 8E 67 0E BD 65 8F 20 05 2C 50
6710  43 52 04 7D 65 48 27 05 86 5D BD 65 6E BE 65 2E
6720  AE 84 8C 52 54 26 02 8D 00 BD 65 81 7A 65 44 27
6730  01 39 86 0C BD 65 6E 86 3C B7 65 44 39 BE 65 2C
6740  BD 66 27 BE 65 3A 8F 65 61 BD 6F AB 25 18 BE 65
6750  5B BF 65 30 BE 65 5D BF 65 32 BE 65 5F BF 65 34
6760  8E 20 20 BF 65 36 20 85 B6 65 2A B7 65 62 7F 65
6770  61 BD 6F AB 25 F0 B6 65 30 81 23 26 D1 86 23 BD
6780  65 6E 20 CA BE 65 2A BF 65 61 20 E5 BD 70 E8 8E
6790  60 06 BD 65 89 BD 65 81 86 3A BD 65 6E BD 65 98
67A0  8E 67 C0 BD 65 6B A1 84 27 0D 30 01 30 01 30 01
67B0  8C 67 E4 26 F1 20 DE BD 65 81 AE 01 AD 84 20 D5
67C0  43 67 EB 41 70 7B 45 70 22 50 68 26 4C 68 E3 53
67D0  68 DC 52 68 B1 5A 70 E8 54 6B 3C 55 68 07 51 F0
67E0  00 48 67 E4 8E 60 40 BD 65 89 39 8E 61 42 BD 65
67F0  89 86 3F BD 65 6E BD 65 9C BD 65 6B 7F 65 54 81
6800  59 26 03 B7 65 54 39 8E 61 5B BD 65 89 BE 70 F8
6810  BF 65 3A 8E 65 3A BD 65 A6 8E 61 62 BD 65 89 8E
6820  70 FA BD 65 A6 39 BE 65 4B BF 65 45 8E 00 00 BF
6830  65 23 86 80 B7 65 57 BD 69 08 24 40 BE 70 F8 8D
6840  2D C6 04 BC 70 FA 27 26 34 04 C6 06 A6 84 BD 65
6850  6E 30 01 5A 26 F6 BD 65 9C BD 65 A6 BD 65 A2 35
6860  04 5A 26 DF 7A 65 44 27 D6 BD 65 81 20 D3 BD 65
6870  81 86 0C BD 65 6E 86 3C B7 65 44 39 7D 65 47 26
6880  0C 7D 65 55 26 18 7D 65 56 26 1E 20 AA BE 65 2C
6890  BD 66 27 BE 65 3A BF 65 61 BD 6F DC 20 99 B6 65
68A0  2A B7 65 62 7F 65 61 20 F0 BE 65 2A BF 65 61 20
68B0  E8 8E 61 5B BD 65 89 BD 66 13 25 F5 BF 65 4B 8E
68C0  61 62 BD 65 89 BD 66 13 25 F5 BF 65 4D 8E 61 69
68D0  BD 65 89 BD 66 13 25 F5 BF 65 52 39 86 80 B7 65
68E0  58 20 06 7F 65 58 BD 68 6E BE 65 4B BF 65 45 8E
68F0  00 00 BF 65 23 BD 65 81 BD 65 81 BD 65 81 8D 08
6900  24 01 39 BD 66 3D 20 F6 B6 65 4E B0 65 46 B6 65
6910  4D B2 65 45 24 01 39 BE 65 45 BF 65 25 B6 65 24
6920  8B 01 19 B7 65 24 B6 65 23 89 00 19 B7 65 23 8E
6930  00 00 BF 65 27 BF 65 2A BF 65 2C BF 65 2E BF 65
6940  66 BF 65 40 7F 65 68 7F 65 47 7F 65 48 7F 65 4F
6950  7F 65 50 7F 65 51 7F 65 29 7F 65 55 7F 65 56 8E
6960  65 30 86 20 C6 08 A7 84 30 01 5A 26 F9 8D 03 1C
6970  FE 39 BE 65 25 A6 84 B7 65 27 81 10 26 03 7E 6F
6980  04 81 11 26 03 7E 6D 81 84 F0 44 44 44 8E 69 A8
6990  BF 65 3A BB 65 3B B7 65 3B 86 00 B9 65 3A B7 65
69A0  3A BE 65 3A AE 84 6E 84 69 F8 6B DD 6C BF 6E 64
69B0  6A 71 6A 85 6B 49 6A 2D 6D 0A 6A 23 6B 3A 6A 67
69C0  6D C7 6A 28 6B 3F 6A 6C B6 65 27 84 0F 48 48 BF
69D0  65 2E BB 65 2F B7 65 2F 86 00 B9 65 2E B7 65 2E
69E0  39 34 02 44 44 44 44 BD 65 D8 A7 84 35 02 BD 65
69F0  D8 30 01 A7 84 30 01 39 8E 63 A3 8D CB BE 65 25
6A00  30 01 A6 84 B7 65 2A 30 01 BF 65 45 86 3C B7 65
6A10  30 86 24 B7 65 31 B6 65 2A 8E 65 32 BD 69 E1 7A
6A20  65 55 39 8E 64 A3 20 D3 8E 64 E3 20 CE 8E 63 A3
6A30  BD 69 C8 BE 65 25 30 01 BF 65 3A AE 84 BF 65 2A
6A40  BE 65 3A 30 01 30 01 BF 65 45 86 3E B7 65 30 86
6A50  24 B7 65 31 8E 65 32 B6 65 2A BD 69 E1 B6 65 2B
6A60  BD 69 E1 7A 65 56 39 8E 64 A3 20 C4 8E 64 E3 20
6A70  BF 86 41 B7 65 31 8E 63 A3 BD 69 C8 BE 65 25 30
6A80  01 BF 65 45 39 86 42 20 EA BE 63 17 84 07 81 04
6A90  27 09 81 05 26 0B 7A 65 67 20 03 7A 65 66 7A 65
6AA0  47 81 07 26 03 7A 65 56 34 02 48 48 48 C6 08 BD
6AB0  6B 7C 35 02 81 03 27 2E 85 01 27 2B BE 65 45 AE
6AC0  84 BF 65 3A BF 65 2A BF 65 2C BE 65 45 30 01 30
6AD0  01 BF 65 45 BD 6C A8 8E 65 31 B6 65 3A BD 69 E1
6AE0  B6 65 3B BD 69 E1 39 BE 65 45 A6 84 B7 65 2D 2A
6AF0  03 7A 65 2C 30 01 BF 65 45 BD 70 FC 8E 65 31 BD
6B00  69 E1 39 7A 65 51 86 24 B7 65 30 86 2C B7 65 33
```

6B10	86	52	B7	65	34	8E	65	34	BF	65	40	7D	65	50	26	15
6B20	B6	65	28	84	1F	85	10	21	08	8E	65	31	BD	69	E1	20
6B30	7D	8A	E0	20	F4	B6	65	29	20	E9	8E	64	A3	20	0D	8E
6B40	64	E3	20	08	B7	65	29	20	1A	8E	63	A3	BD	69	C8	BE
6B50	65	25	30	01	A6	84	30	01	BF	65	45	7D	65	50	26	E4
6B60	B7	65	28	2A	9E	85	10	27	03	7A	65	48	85	08	27	03
6B70	7E	6A	89	84	07	48	48	8E	62	F7	C6	04	BF	65	3A	BB
6B80	65	3B	B7	65	[illegible]	86	00	B9	65	3A	B7	65	3A	8E	65	30
6B90	BF	65	3E	BE	65	3A	A6	84	30	01	BF	65	3A	BE	65	3E
6BA0	81	52	26	03	BF	65	40	A7	84	30	01	5A	26	E2	7D	65
6BB0	50	26	19	B6	65	28	84	60	27	17	80	20	27	17	80	20
6BC0	27	17	86	53	BE	65	40	27	02	A7	84	39	B6	65	29	20
6BD0	E5	86	58	20	EF	86	59	20	EB	86	55	20	E7	8E	63	E3
6BE0	BD	69	C8	BE	65	25	30	01	BF	65	45	B6	65	27	84	0F
6BF0	81	09	27	1C	81	0D	27	18	81	06	27	15	81	07	27	14
6C00	81	0A	27	13	81	0C	27	0F	81	0E	27	29	81	0F	27	25
6C10	39	7E	6C	72	7E	6C	72	BE	65	45	A6	84	30	01	BF	65
6C20	45	B7	65	2A	8E	65	32	BD	69	E1	86	23	B7	65	30	86
6C30	24	B7	65	31	39	BE	65	45	A6	84	30	01	BF	65	45	B7
6C40	65	2A	8E	63	57	84	F0	27	08	30	01	30	01	80	10	20
6C50	F6	AE	84	BF	65	30	86	2C	B7	65	32	B6	65	2A	8E	63
6C60	57	84	0F	27	07	30	01	30	01	4A	20	F7	AE	84	BF	65
6C70	33	39	7A	65	47	BD	6D	6A	BE	65	45	AE	84	BF	65	2A
6C80	BF	65	2C	BE	65	45	30	01	30	01	BF	65	45	8E	65	33
6C90	B6	65	2A	F6	65	2B	7D	65	4F	26	20	CB	03	89	00	BD
6CA0	69	E1	1F	98	4D	BD	69	E1	B6	65	46	BB	65	2D	B7	65
6CB0	2D	B6	65	45	B9	65	2C	B7	65	2C	39	CB	04	20	DE	8E
6CC0	64	23	BD	69	C8	20	06	8E	63	77	BF	65	2E	7A	65	47
6CD0	BE	65	25	30	01	A6	84	30	01	BF	65	45	B7	65	2A	7F
6CE0	65	2C	B7	65	2D	2A	03	7A	65	2C	BD	6D	6A	8E	65	33
6CF0	B6	65	2A	BB	02	2B	05	BD	69	E1	20	AC	34	02	86	FF
6D00	BD	69	E1	35	02	BD	69	E1	20	9E	B6	65	27	84	0F	81
6D10	0D	27	B4	8E	64	A3	BD	69	C8	BD	6C	2A	B6	65	27	84
6D20	0F	81	0C	24	40	81	03	27	3C	7A	65	55	BE	65	25	30
6D30	01	A6	84	30	01	B7	65	2A	7D	65	4F	27	0D	A6	84	B7
6D40	65	2B	7A	65	56	7F	65	55	30	01	BF	65	45	8E	65	32
6D50	B6	65	2A	BD	69	E1	7D	65	4F	27	06	B6	65	2B	BD	69
6D60	E1	7F	65	4F	39	7A	65	4F	20	BF	86	2A	B7	65	30	86
6D70	2B	B7	65	31	86	24	B7	65	32	39	8E	63	7F	BF	65	2E
6D80	39	BE	65	25	30	01	A6	84	30	01	BF	65	45	B7	65	28
6D90	81	3F	27	E6	85	08	26	1B	8E	63	83	BF	65	2E	B6	65
6DA0	28	84	30	27	2B	80	10	27	0F	80	10	27	11	BE	65	45
6DB0	7E	6A	38	8E	63	87	20	E3	BE	65	45	7E	6A	02	BE	65
6DC0	45	7A	65	50	7E	6B	54	8E	64	E3	BD	69	C8	7E	6D	19
6DD0	BE	65	45	AE	84	BF	65	2A	BE	65	45	30	01	30	01	BF
6DE0	65	45	BD	6C	2A	8E	65	32	B6	65	2A	BD	69	E1	B6	65
6DF0	2B	BD	69	E1	39	7E	6E	7C	BE	65	25	30	01	A6	84	2B
6E00	F4	B7	65	2A	30	01	BF	65	45	8E	00	00	BF	65	3A	48
6E10	B7	65	3B	78	65	3B	79	65	3A	BB	65	3B	B7	65	3B	86
6E20	00	B9	65	3A	B7	65	3A	8E	61	77	BF	65	3C	B6	65	3B
6E30	BB	65	3D	B7	65	3B	B6	65	3C	B9	65	3A	B7	65	3A	BE
6E40	65	3A	AE	84	BF	65	30	BE	65	3A	AE	02	BF	65	32	BE
6E50	65	3A	AE	04	BF	65	34	8E	20	20	BF	65	36	8E	61	73
6E60	BF	65	2E	39	8E	64	63	BD	69	C8	B6	65	27	85	08	27
6E70	14	7D	65	54	27	06	81	3F	10	27	FF	7C	BE	65	25	30
6E80	01	BF	65	45	39	85	04	26	08	BE	65	25	30	01	7E	6B
6E90	54	BE	65	25	30	01	A6	84	30	01	BF	65	45	B7	65	2A
6EA0	8E	65	30	85	80	27	06	C6	50	E7	84	30	01	85	40	27
6EB0	11	F6	65	27	C4	02	27	04	C6	53	20	02	C6	55	E7	84
6EC0	30	01	85	20	27	06	C6	59	E7	84	30	01	85	10	27	06
6ED0	C6	58	E7	84	30	01	85	08	27	06	C6	44	E7	84	30	01
6EE0	85	04	27	06	C6	42	E7	84	30	01	85	02	27	06	C6	41
6EF0	E7	84	30	01	85	01	27	04	C6	43	E7	84	39	8E	63	7B
6F00	BF	65	2E	39	BE	65	25	30	01	A6	84	30	01	BF	65	45
6F10	B7	65	28	81	3F	27	E6	85	80	26	1E	8E	64	23	B6	65
6F20	28	BD	69	C8	7A	65	68	7E	6C	72	8E	63	8B	20	1F	8E
6F30	63	8F	20	1A	8E	63	93	20	15	85	40	26	14	84	0F	81
6F40	03	27	E7	81	0C	27	E8	81	0E	27	E9	8E	63	97	7E	6D
6F50	9B	85	01	27	05	8E	63	9F	20	F4	8E	63	9B	20	EF	BE
6F60	70	F8	BC	70	FA	27	2D	BF	65	59	AE	84	BC	65	5B	26
6F70	1F	BE	65	59	AE	02	BC	65	5D	26	15	BE	65	59	AE	04

```
6F80  BC 65 5F 26 0B BE 65 59 AE 06 BF 65 61 1C FE 39
6F90  8D 05 20 CE 1A 01 39 B6 65 5A 8B 08 B7 65 5A B6
6FA0  65 59 89 00 B7 65 59 BE 65 59 39 BE 70 FB BC 70
6FB0  FA 27 E1 BF 65 59 AE 06 BC 65 61 27 04 8D DB 20
6FC0  ED BE 65 59 AE 84 BF 65 5B BE 65 59 AE 02 BF 65
6FD0  5D BE 65 59 AE 04 BF 65 5F 1C FE 39 8D CD 24 41
6FE0  BE 70 FA 86 2E A7 84 30 01 B6 65 63 BD 69 E1 B6
6FF0  65 64 BD 69 E1 B6 65 64 8B 01 19 B7 65 64 B6 65
7000  63 89 00 19 B7 65 63 86 30 A7 84 B6 65 61 A7 01
7010  B6 65 62 A7 02 BE 70 FA BF 65 59 BD 6F 97 BF 70
7020  FA 39 BD 65 81 86 24 BD 65 6E BD 66 13 25 F3 BF
7030  65 61 BD 6F AB 25 38 BD 65 98 8E 65 5B C6 06 A6
7040  84 BD 65 6E 30 01 5A 26 F6 BD 65 98 BE 65 59 C6
7050  06 86 20 A7 84 30 01 5A 26 F9 BE 65 59 C6 06 BD
7060  65 6B 81 0D 27 0B 81 21 25 F5 A7 84 30 01 5A 26
7070  EE 39 86 3F BD 65 6E BD 65 81 39 BD 65 81 86 24
7080  BD 65 6E BD 66 13 24 01 39 BF 65 61 BD 6F AB 24
7090  A6 BD 65 98 8E 65 5B 86 20 A7 84 30 01 8C 65 61
70A0  26 F7 8E 65 5B BD 65 6B 81 21 25 09 A7 84 30 01
70B0  8C 65 61 26 F0 B6 65 5B 81 20 27 05 BD 6F 5F 25
70C0  03 7E 70 72 8D 02 20 83 8E 65 5B BF 65 3A C6 08
70D0  BE 65 3A A6 84 30 01 BF 65 3A BE 70 FA A7 84 30
70E0  01 BF 70 FA 5A 26 E9 39 8E 80 00 BF 70 FB BF 70
70F0  FA 8E 00 00 BF 65 63 39 00 00 00 00 34 02 BD 6C
7100  A8 35 02 39 00 7F 71 04 BC 65 45 27 32 7D 65 48
7110  26 15 7D 65 55 26 18 7D 65 56 26 16 B6 65 30 81
7120  23 27 0F 81 2A 27 08 BD 65 9C BD 65 B0 20 03 7C
7130  71 04 BD 65 9C BC 65 45 27 05 BD 65 B0 20 F6 C6
7140  18 39 A6 80 81 20 27 11 BD 65 6E 81 24 26 0A 7D
7150  71 04 27 05 7A 71 04 27 07 8C 65 38 26 E4 20 E1
7160  FE 65 25 A6 C4 81 10 26 D9 FC 65 45 B3 65 25 E3
7170  42 FD 65 3A 8E 65 3A 7E 65 A6
```

当移植到MB-6890系统中时，需要进行以下的修改和补充：原来的程序是以打印机输出为前提的，仅限于显示画面的输出，在代替页面（画面清除）之前，要等待着一个字符键的输入才行。在开始执行命令之前，因为还要等待键盘输入，所以如果想令其停下来，请按下任何一个键都可以。修改和补充增加后的检查和为$063888（$6000～$7189）。其修改和补充部分如下：

地址	新数据
6544	18
656C	71 7A
657A	71 7F
6735	71 84
6738	18
67DF	FD A9
6874	71 84
6877	18
70E9	14 00
717A	0F 9E BD E8 04 0F
7180	9E 7E E8 20 8D F4 86 0C 20 F5

附录1 莫托罗拉公司的汇编程序

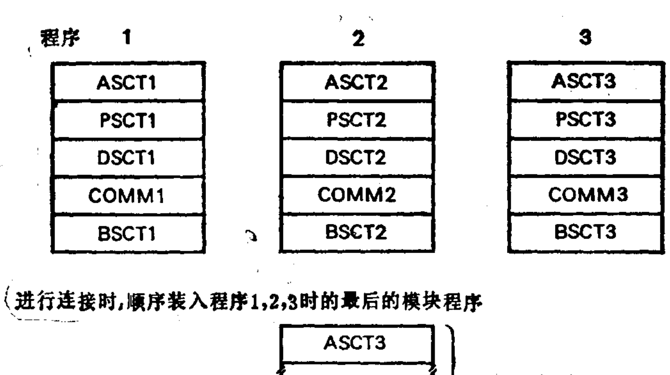
MC6809使用的汇编程序已经有几个公司在销售，各个公司的记忆符号都是根据MC6809的数据手册、使用相同的表示方法。在汇编程序中的命令也用同样的表示方法和命令码（RMB，FCB，FDB等）。这里只对MOTOROLA公司的宏汇编程序进行说明。需要注意的是，在这些宏汇编命令中，有其它公司在汇编程序中没有用到的命令。

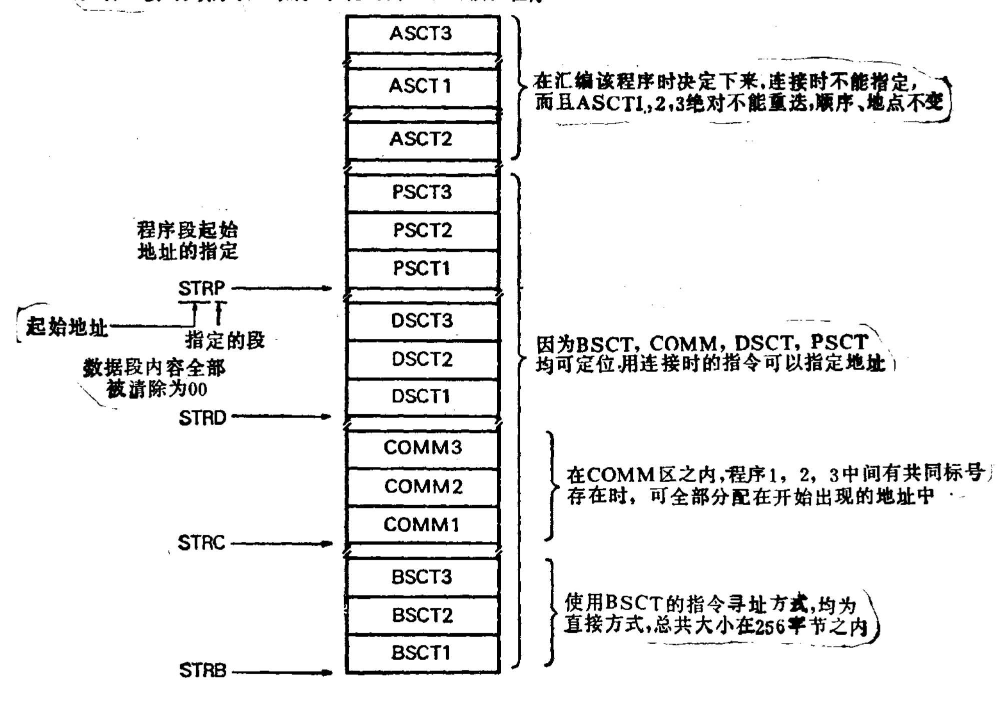
（1）按字母顺序排的宏汇编程序的伪指令

伪指令	说明
ASCI	分配绝对地址区
BSCI	分配页面区
BSZ	保证清除数据区
COMM	分配共用区
CSCT	在共用区分配清除的状态
DSCT	分配数据区
END	程序结束
ENDC	条件汇编程序结束
ENDM	宏程序结束
EQU	分配给符号以数值
FAIL	表示由程序人员造成的错误
FCB	分配字节常数
FCC	分配字符序列
FDB	分配2字节常数
IDNT	表示可定位代码的识别
IFEQ	条件汇编（= 0）
IFGE	条件汇编（>= 0）
IFGT	条件汇编（> 0）
IFLE	条件汇编（<= 0）
IFLT	条件汇编（< 0）
IFNE	条件汇编（≠ 0）
MACR	宏定义开始
NAM	定义程序名称
OPT	指定汇编程序输出方式选择
ORG	程序计数器置数
PAGE	替换程序清单的页面
PSCT	分配程序区
REG	指定寄存器列表
RMB	保证存储区
SET	指定符号的数值
SETDP	指定直接页面寄存器的数值
SPC	传送程序清单
TTL	指定列表清单的标题
XDEF	指定输出到其它程序的符号
XREF	从其它程序来的符号
! ∧	幂〔数据 =（数据）*（操作数）〕
! ·	"与"逻辑符号
! +	"或"逻辑符号
! ×	"异或"逻辑符号
! <	左移n位
! >	右移n位
! L	循环左移n位
! R	循环右移n位

如果在MC6809汇编程序中规定了可选项REL（OPT RET）时，则输出为可定位的程序，需要使用绝对地址寻址的操作数，可分配给假独立变数。根据宏汇编程序，对应假独立变数输出符号表的数据，这些数据由连接装入程序变换为绝对地址。

在许多个分割的程序中，如果用REL可选项对它们一个个地进行汇编时，用连接装入程序，可以把分割的程序合为一个程序。例如把1，2，3程序合为一个程序时，各个程序的段落再配置如附图1.1所示。

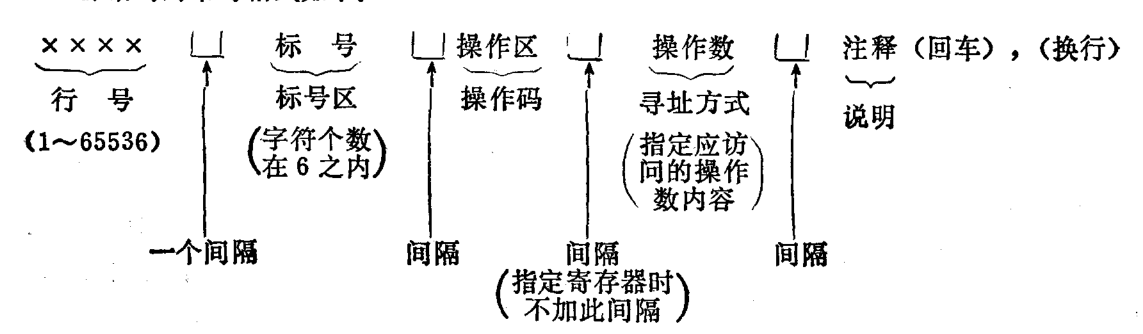




附图1.1

（2）源语句的书写格式

源语句的书写格式如下：



行号从1～16位长，最大数可到65536，还可不加行号。如果开始行未设行号，则汇编时，从下一行到最后一行都按无行号来执行。若中途又出现了行号，则认为语法有错。

行号后设一个字符的间隔（ASCII 码数据$20）。

由换行后第一个字符（第一列）写*来定义解释行。

在包含行号的源程序中，在行号后设置一个字符的间隔，第二个字符写 * 。

```
例:  100 |_| * COMMENT
          ↑
        一间隔

     200 |_|  |_| ASLAS
          ↑    ↑
          二 间 隔
```

标号在 6 个字符之内。不设行号时，标号要从第一列书写起。在第一个字符中，不能使用下列字符：

* （定义注释行的）；

#，$，@，%（指定数据进位制的）；

A，B，X，Y，CC，D，DP，PC，PCR，S，U（指定寻址方式的）。

有行号时，行号和标号之间必须设一间隔。

```
例:  200 |_| TEST  ASLA      行号为200
          ↑
       一个间隔              标号为TEST（测试）

    200 |_||_| TEST ASLA
         ↑  ↑
    二个间隔将出现错误
```

标号是可以作为操作数来使用的，如

```
     DATA EQU $100
          LDX #DATA
```

在汇编过程中，这两行将被展开为

```
     LDX # $100
```

操作码区需要接在各个记忆符之后。规定A，B，D，X，Y，CC，S，U等寄存器时，则不要再输入间隔。在MC6809汇编程序中，和MC6800的汇编程序不同，在指定寄存器时，如果输入间隔，将出现错误。例如把ADDA打成ADD |_| A，就会出现错误。

除去累加器偏移变址方式的起始字符外，操作数皆要规定寻址方式。

在操作数中，进行数值或逻辑运算时，在内部只分配16位长的整数区。对于16位数据的溢出，不进行任何处理。相反，这种功能只是在要求数据的高位字节、低位字节时使用的。例如，在DATA/256* 256中，只有高位字节中有数，而低位字节为 0 0 ，需要低位字节时，则为DATA* 256/256。高位字节的数值用作置位DP时是方便的，例如

```
DATA    EQU    $2345
DPSET SETDP  DATA/256
```

DP被置为23。

在操作数中，用下列符号：

* 乘法

/ 除法

+ 加法

- 减法

操作码为分支转转移指令、跳越转移指令等时，进行程序计数器的处理中，操作数内的*表示自己的操作码的起始地址。例如：BRA* 和LOOP BRA LOOP是一样的，将成为永远的循环状态。

源程序一般用名称（NAM）开始，用END结束。

```
××××  ␣  NAM AAAAAA        程序或模块之名称限6个字符之内。
××××  ␣  TTL  YYYYYY…Y     在程序表格清单上面打印标题。
××××  ␣  OPL  ZZ, ZZ       规定可选项目。
              ⋮             程序。
××××  ␣  END START         程序结束。
```

如果规定END的操作数为执行行起始地址的标号，则在程序装入时，可做到不规定执行地址。在S格式中，

```
S903     ××××      Z Z
       执行起始地址   ↑
                   检查和
```

用××××表示的装入结束之后，可以自动地启动。

（3）宏定义的方法

宏定义时，要把被定义的名称××××××（6个字符之内）写在标号区域之内，在记忆区，从写有“MACR”的行开始执行，用ENDM来结束。在宏名称中，已有的记忆符（符号指令字）和宏指令已经定义了的名称都不能使用。因为宏指令是可以嵌套工作的，所以，在宏指令的内部可以使用其它宏指令。

```
××××××   MACR
            ⋮
         ENDM
```

有条件的宏指令定义如下：

```
CHECK MACR
      IFNC  \0, ALPH
      IFNC  \0, BATA
      FALL  * INVALID ARGUMENT *
      ENDC
      ENDC
      IFC   \0, ALPH
      LEAX  ·ALPH, PCR
      ENDC
      IFC   \0, BATA
```

```
        LEAX   ·BATA, PCR
        ENDC
        ENDM
主程序
   ⋮
```

CHECK DATA→变数＼0因为没有配合好,所以打印 *INVALID ARGUMENT *

CHECK ALPH→LEAX·ALPH, PCR 被汇编

条件汇编的清单必须用ENDC来结束，而且条件汇编指令的数目和ENDC要一一对应，因此两者数目相等。

注意：程序中的"＼"在JIS编码（日工业标准）中为"￥"。

附录2 S格式的记录

在MC6800中的S格式记录，用在MC6809系统中，当有2M字节地址时，已经不能进行表示了。因此，对S格式的记录，要重新进行，并作如下规定：

（1） 使用字符

数目字0～9，字母A～F，$0D（CR），$0A（LF），$00（NULL）

（2） 记录形式

S0 数据集名称

S1 16位地址数据记录

S2 24位地址数据记录

S3 32位地址数据记录

S5 发出数据记录数记录

S7 32位地址数据结束并执行地址记录

S8 24位地址数据结束并执行地址记录

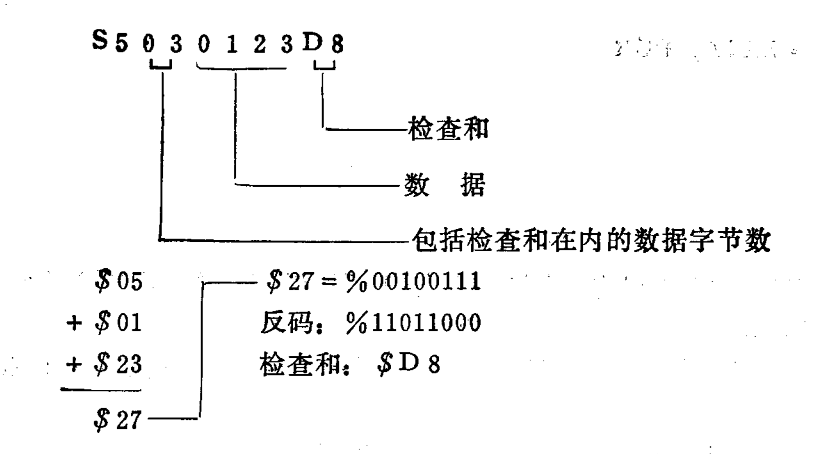
S9 16位地址数据结束并执行地址记录

S4，S6 当前还未使用

S5是在S0所开始的数据块之中，是为正确读入或判别而使用的格式。在MC6800系统中，只用S0，S1，S9，在没有读完一个数据记录块时，是不能进行错误判别的。在MC6809系统中，数据块的数目也可确定，即使是24位或32位长的地址，也可相应有记录格式。使用存储器管理单元时，则需要保证读出开始的数据记录，并且其中要写有：程序的基本段、数据段、共用段、绝对段的起始地址及其区域的大小。

S_4，S_6是当前还未定义的记录，可以作为各段的说明、保证数据记录的条件等使用。不管数据记录的种类如何，S格式的检验码总是把一连串的S0，S1……的记录，变为8位数据后，全部进行相加，再把由此得到的8位数据的反码作为检查和。

检查和的计算如下例所示：



附录3 维修设备和维修方法实例

维修MC6809系统时，只有示波器等设备是不能解决问题的。其原因是，在一条数据/地址线上的波形，所能看到的也只是多重化的不规则脉冲。即使是采用测试程序（假定测试程序用20个机器周期作为一个大测试周期，逻辑值 1， 0 也可读取），但用这样的程序维修整个系统也是不可能的。为了解决这种问题，可以使用 HP 公司的符号分析法逻辑电路测试仪。有关其使用方法，可以参考HP公司的应用资料（AN03－3）。

符号分析法（Signature Analysis)

在微机系统中，如果增加自诊断逻辑电路来判别哪个部分不良，那么势必提高成本，一般是不这么办的。象大型计算机那样的处理机，本身发现自己的故障，作应急处理的手段，在微机中也是不可能的。但是，只有一个部件不良，就把整个系统送回工厂，从工厂换插件板，这样，往返的成本和时间都会增加许多。因此，为了维护全部可更换的插件板，最好的办法是减少部件。用符号分析法解决这类问题，可以得到大体上满意的结果。在 HP 公司的5004符号分析器中，设有相应于实际维修插件板的对照结果和具体说明。

为了维修插件电路板，要使板上固有的数据总线与系统脱开，编入电阻跳接器。如果这时所编入的内容是在处理机总清之后，执行 1 字节指令CLRB（$ 5 F)，那么处理机总清后，即执行 $ 5 F 5 F 地址的 $ 5 F (CLRB) 。由于执行了这个一字节指令，各地址线正好是按二进制计数器计数。因此，如果在各地址线上的集成电路、大规模集成电路芯片上的选片端测试输入地址，则地址的关系情况，可以全部测试出来。这时，符号分析器的时钟信号在处理机的E脉冲下降沿处为有效，分析的起始和停止范围在A 15的下降沿处为有效。首先测试处理机地址A 0， A 1， ……， A 15的符号。接着进行各个板上的选片测试。如果没有问题，由所对应选片的起始和终止范围，检查程序ROM的内容。

处理机和地址线以及ROM的测试，如果都合格，那么，不使用 RAM 的测试程序应该可以执行。因此，如果在ROM中含有RAM或I/O测试程序，可先对RAM或I/O进行测试。

微机系统中所用的硬逻辑，几乎就是地址和选片线路。实际上，在任何运行方式中，不良的部分几乎都可发现。当输入结构合格时，如果输出结果不良，这可能是由于输出门不好，或者是作为其输入门的输入部分不好，因此，在这种情况下，更换部件就会变好。下面讲述一下维修MC6809系统的实例。

地址线的测试

起始　A 15下降沿

终止　A 15下降沿

时钟　E　下降沿

保持/测试　截止

把测试笔接到V_{SS}，V_{DD}上，并点亮控制灯，以确定没有不稳定状态（Unstable）出现。这时，MC6809地址线的代码为下列状态（但是5004 A符号分析器用四位十六进制显示。为了保证七段LED易于读出，所以使用 0 1 2 3 4 5 6 7 8 9 A C F H P U 的字符集进行判断）：

VSS	0000	A4	1U5p (1F5E)	A10	37C5 (37B5)
VDD	0003	A5	0356	A11	6U28 (6F28)
A0	UUUU (FFFF)	A6	U759 (F759)	A12	4FCA (4CBA)
A1	FFFF (CCCC)	A7	6F9A (6C9A)	A13	4868
A2	8484	A8	7791	A14	9Up1 (9FE1)
A3	p763 (E763)	A9	6321	A15	0001

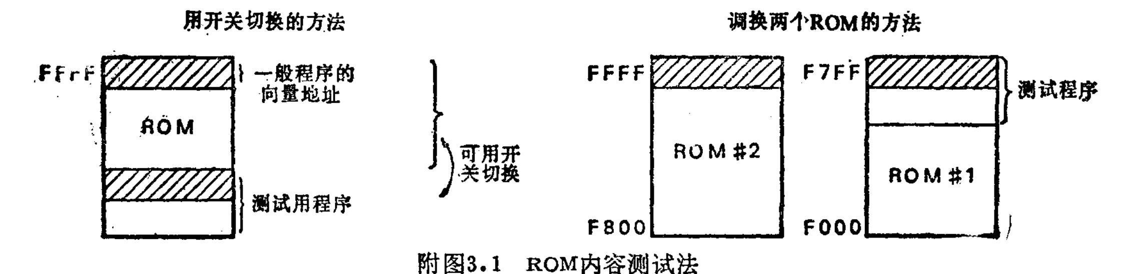
这些数据在MC6809，MC6800，MC 6801/3 中也都相同，如果逻辑编码处在不稳定情况下，则可以用示波器来确定E脉冲是否稳定。也可以在特定地址上，用示波器观测输出波形，寻找异常原因。

其次，要测试地址译码器的输出。在用 ROM 作的译码器上设有负逻辑（$\overline{CE}$）的启动输入端。当地址是从＄F000到＄F3FF的 1 K字节时（用2708），CE为＄EC2C。地址从＄F400到＄F7FF时，CE为＄8C16。这样，全部的输出可以使用一条线实现，如果逻辑电路中没有多余的部分，则全部输出可以按不同的符号编码。通常，是采用不同的符号编码，若是使用同一符号的编码，这时要看是固定性连接、还是锡焊软连接等情况来进行判别。

同样，如果测试全部地址译码器是好的，下面就该测试 ROM 的数据。这时信号的起始和终止点是：当使用的ROM选片信号为负逻辑时，起始用下降沿触发，终止用上升沿触发；反之，当用正逻辑时，起始用上升沿触发，终止用下降沿触发。

用测试笔可以对各个数据的输出、数据总线、处理器的数据总线控制的外部芯片的数据等所有的数据总线进行测试。根据测试结果，即可以确定数据总线驱动器（ROM这一端）和数据总线接收器（MPU这一端）的好坏。

当认为所有数据总线的测试结果是良好的时候，则可开始对ROM内的数据进行测试。执行测试程序时，当有二个以上的 ROM 程序存在时，可用开关切换分析程序， 也可以每次调换ROM进行测试（为附图3.1所示）。在HP公司的应用手册中，是用开关进行两个ROM分别调换的方式，其测试用的逻辑电路可省。但是，当用开关方式时，测试程序的长度要完全固



附图3.1　ROM内容测试法

定才行，这是一个缺点。

调换二个ROM的方法

在一般的执行中，ROM＃2中包括有程序本身用的全部的向量。ROM＃1中的高位区存储测试程序。测试时，调换ROM＃1和ROM＃2之后，从ROM＃1的Reset向量地址开始测试。由于系统构成的不同，测试程序的内容将有很大的区别。在开始测试时，因为RAM的工作还未确定好坏，所以等高速缓冲存储区的RAM、用户区的RAM以及对一般的RAM测试结束之后，才可开始调用子程序或往RAM中传送数据。

最后进行输入输出程序和中断处理的测试。

中断处理的测试，一般需要较难的输入输出手续。在没有自动化的情况下，从键盘等输入设备来的信号还需要靠人工来进行。

更详细的方法，或者有关符号分析法的理论，请阅读HP公司的应用资料AN03-3和5004A的符号分析器使用指南。

附录4　JIS编码表(C6220)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	(SP)	0	@	P	`	p				ー	タ	ミ		
1	SOH	DC_1	!	1	A	Q	a	q			。	ア	チ	ム		
2	STX	DC_2	"	2	B	R	b	r			「	イ	ツ	メ		
3	ETX	DC_3	#	3	C	S	c	s			」	ウ	テ	モ		
4	EOT	DC_4	$	4	D	T	d	t			、	エ	ト	ヤ		
5	ENQ	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	BEL	ETB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8	BS	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF	SUB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
B	VT	ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	FF	FS	,	<	L	¥	l	|			ャ	シ	フ	ワ		
D	CR	GS	−	=	M	]	m	}			ュ	ス	ヘ	ン		
E	SO	RS	.	>	N	^	n	‾			ョ	セ	ホ	゛		
F	SI	US	/	?	O	_	o	DEL			ッ	ソ	マ	゜		

附录5　6809指令系统一览表

附录5中用下列代号

寄存器：

A，B	累加器
D	双累加器（A和B连接起来，A放高位）
DP	直接页面寄存器
X，Y	变址寄存器
PC	程序计数器
S	硬件堆栈指示器
U	用户堆栈指示器
CC	状态码寄存器

状态位，状态码寄存器0位～7位：

C	进位（借位）位
V	溢出位
Z	零标志位
N	符号（负号）位
I	（规则的）中断屏蔽位
H	半进位位
F	快速中断屏蔽位
E	全部标志位

状态标志栏代号：

(blank)	操作不起作用标志
X	操作起作用标志
0	清零操作标志
1	置位操作标志

其他代号和缩写词：

AC_x	累加器A或B中任一个累加器
adr8	8位地址，一字节参数，它可用作基本（直接）页面直接地址存储单元
adr16	16位存储器地址
b_0-b_7	后缀字节或8位寄存器的各位
C	进位标志数，0或1
data 8	8位二进制数单元
data 16	16位二进制数单元
disp 8	8位二进制地址移动标记
disp 16	16位二进制地址移动标记
EA	由任一寻址方式计算出的有效地址
M	用基本页面直接、直接扩展、变址或者间址寻址确定的存储器地址

〔M〕	M的内容
〔M〕:〔M+1〕	16位地址单元内的内容，其高位字节码是M的内容，其低位字节码是下一个高位地址的内容
reg	16位变址寄存器或堆栈指示器（S，U，X或Y）
reg, list	存入堆栈或者由堆栈中恢复过来的寄存器表格
R16	16位寄存器（D，S，U，X或Y）
R1,R2	双8位或双16位的两个寄存器
SP	堆栈指示器（S或U）
ind torms	附录6中讲到的变址或间址寻址方式的任一种
〔interrupt vector〕	某个中断向量地址中的内容
××（H1）	16位分量××中高8位码
××（L0）	16位分量××中低8位码
〔 〕	存储单元内容用括号括起来
〔〔 〕〕	隐含存储器地址：由寄存器内容指定的存储单元的内容
∧或∩	逻辑与
∨或∪	逻辑（加）或
∀或⊕	逻辑异或
←	按箭头传递数据
← →	双向同时传送数据：源、目内容交换

6809指令系统一览表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
													6809所有基本存储器基本指令有下列寻址方式：基本页面直接寻址，直接扩展寻址，変址寻址，间接寻址。
基本存储器基本指令和I/O指令	LDA LDB	adr8 adr16 ind,forms	2 2 2+	4 5 4+					×	×	0		$(AC_X) \leftarrow (M)$ 把指定存储器内容装入到累加器A或B中
	STA STB	adr8 adr16 ind,forms	2 3 2+	4 5 4+					×	×	0		$(M) \leftarrow (AC_X)$ 把累加器A或B的内容存到指定的存储器中
	LDD	adr8 adr16 ind,forms	2 3 2+	5 6 5+					×	×	0		$(D) \leftarrow (M):(M+1)$ 把指定存储器内容装到双字长累加器中，符号位N取此数第15位的值(即累加器A第七位值)
	STD	adr8 adr16 ind,forms	2 3 2+	5 6 5+					×	×	0		$(M):(M+1) \leftarrow (D)$ 把双累加器内容存到指定的存储器中，符号位取数据第15位的值(累加器A第七位的值)
	LDX LDU	adr8 adr16 ind,forms	2 3 2+	5 6 5+					×	×	0		$(reg) \leftarrow (M):(M+1)$ 把存储器内容装到指定的寄存器(X,Y,U或S)中，符号位N取此数据第15位值
	LDY LDS	adr8 adr16 ind,forms	3 4 3+	6 7 6+					×	×	0		
	STX STU	adr8 adr16 ind,forms	2 3 2+	5 6 5+					×	×	0		$(M):(M+1) \leftarrow (reg)$ 把指定寄存器的内容存到存储器中，符号位N取此寄存器第15位的值
	STY STS	adr8 adr16 ind,forms	3 4 3+	6 7 6+					×	×	×		

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
扩充存储器基本指令（存储器操作）													6809所有扩充存储器基本指令有下列寻址方式：基本页面直接寻址，直接扩充寻址，变址寻址，间接寻址。
	ADCA ADCB	adr8 adr16 ind,forms	2 3 2+	4 5 4+			×		×	×	×	×	$[AC_X] \leftarrow [AC_X]+[M]+C$ 带进位位加到累加器A或B中
	ADDA ADDB	adr8 adr16 ind,forms	2 3 2+	4 5 4+			×		×	×	×	×	$[AC_X] \leftarrow [AC_X]+[M]$ 把指定的存储单元的内容加到累加器A或B中
	ADDD	adr8 adr16 ind,forms	2 3 2+	6 7 6+					×	×	×	×	$[D] \leftarrow [D]+[M]:[M+1]$ 把存储器16位数值加到双累加器中，操作数的高位字节是指定存储单元内容，操作数低位字节是下一个高位地址存储单元内容
	ANDA ANDB	adr8 adr16 ind,forms	2 2 2+	4 5 4+					×	×	0		$[AC_X] \leftarrow [AC_X] \wedge [M]$ 指定存储单元内容和累加器A或B的内容相"与"
	BITA BITB	adr8 adr16	2 3 2+	4 5 4+					×	×	0		$[AC_X] \wedge [M]$ 指定存储单元内容和累加器A或B的内容相"与" 只影响状态寄存器
	CMPA CMPB	adr8 adr16 ind,forms	2 3 2+	4 5 4+			×		×	×	×	×	$[AC_X]-[M]$ 把指定存储单元内容和累加器A或B内容相比较，只影响状态寄存器
	CMPD	adr8 adr16 ind,forms	3 4 3+	7 8 7+					×	×	×	×	$[D]-[M]:[M+1]$ 16位数据和双累加器比较，只影响状态寄存器，数据高位字节是指定存储单元内容，数据低位字节是下一个高位地址存储单元的内容。
	CMPS CMPU CMPY	adr8 adr16 ind,forms	3 4 3+	7 8 7+					×	×	×	×	$[reg]-[M]:[M+1]$ 16位数据和指定寄存器(S,U,X,Y)内容比较，只影响状态寄存器，数据的高位字节是指定存储单元内容，数据的低位字节是下一高位地址存储单元内容。

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
扩充存储器基本指令(存储器操作)	CMPX	adr8 adr16 ind, forms	2 3 2+	6 7 6+					×	×	×	×	与CMPS/CMPU/CMPY相同
	EORA EORB	adr8 adr16 ind, forms	2 3 2+	4 5 4+					×	×	0		[AC_X]←[AC_X]⊻[M] 指定存储单元内容和累加器A或B内容逻辑异或
	ORA ORB	adr8 adr16 ind, forms	2 3 2+	4 5 4+					×	×	0		[AC_X]←[AC_X]∨[M] 指定存储单元内容和累加器A或B内容逻辑或
	SBCA SBCB	adr8 adr16 ind, forms	2 3 2+	4 5 4+			×		×	×	×	×	[AC_X]←[AC_X]−[M]−C 从累加器A或B中减去指定存储单元内容和进位位内容
	SUBD	adr8 adr16 ind, forms	2 3 2+	6 7 6+					×	×	×	×	[D]←[D]−[M]:[M+1] 从双累加器中减去存储器中16位数值，操作数高位字节是指定存储单元地址中内容，操作数低位字节是下一个高位地址存储单元中内容。
	SUBA SUBB	adr8 adr16 ind, forms	3 2 2+	4 5 4+			×		×	×	×	×	从累加器A或B中减去指定存储单元的内容[AC_X]←[AC_X]−[M]
	ASL	adr8 adr16 ind, forms	2 3 2+	6 7 6+			×		×	×	×	×	C ← [7 ← 0] ←0, V←N⊻C (M) 算术左移，第0位置"0"
	ASR	adr8 adr16 ind, forms	2 3 2+	6 7 6+			×		×	×		×	[7 → 0] → C (M) 算术右移，第7位保留不变
	CLR	adr8 adr16 ind, forms	2 3 2+	6 7 6+					0	1	0	0	[M]←00_{16} 清除指定存储单元

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
扩充存储器基本指令（存储器操作）	COM	adr8 adr16 ind,forms	2 3 2+	6 7 6+					×	×	0	1	〔M〕←〔$\overline{M}$〕 把存储单元内容取1的补码(把存储单元内容求反)
	DEC	adr8 adr16 ind,forms	2 3 2+	6 7 6+					×	×	×		〔M〕←〔M〕−1 存储单元内容减1
	INC	adr8 adr16 ind,forms	2 3 2+	6 7 6+					×	×	×		〔M〕←〔M〕+1 存储单元内容加1
	LSL	adr8 adr16 ind,forms	2 3 2+	6 7 6+			×		×	×	×	×	C ← 7←—0 ←0, V←N∀C (M) 逻辑左移，同ASL
	LSR	adr8 adr16 ind,forms	2 3 2+	6 7 6+					0	×		×	0→ 7—→0 → C (M) 逻辑右移，第七位置"0"
	NEG	adr8 adr16 ind,forms	2 3 2+	6 7 6+			×		×	×	×	×	〔M〕←00_{16}−〔M〕 存储单元内容取2的补码，其结果若为00_{16}，置进位位，否则清除进位位，如其结果为80_{16}，置溢出位，否则清除溢出位
	ROL	adr8 adr16 ind,forms	2 3 2+	6 7 6+					×	×	×	×	C ← 7←—0 ← V=N∀C (M) 存储单元内容带进位位循环左移
	ROR	adr8 adr16 ind,forms	2 3 2+	6 7 6+					×	×	×	×	→ C → 7—→0 (M) 存储单元内容带进位位循环右移

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
扩充存储器基本指令(存储器操作)	TST	adr8 adr16 ind, forms	2 3 2+	6 7 6+					×	×	0		[M] − 00_{16} 测试存储单元内容是否为0或负值
立即数装入指令	LDA LDB	data8	2	2					×	×	0		[AC_X]←data8 装立即数到累加器A或B中
	LDD	data16	3	3					×	×	0		[D]←data16 装立即数到双累加器中，符号位为数据的第15位(A累加器第7位)
	LDU LDX	data16	3	3					×	×	0		[reg]←data16
	LDS LDY	data16	4	4					×	×	0		装立即数到指定寄存器中(X，Y，U或S中)，符号位N为寄存器内容的第15位
立即数操作指令(立即数运算指令)	ADCA ADCB	data8	2	2			×		×	×	×	×	[AC_X]←[AC_X]+data8+C 把立即数和进位位加到累加器A或B中
	ADDA ADDB	data8	2	2			×		×	×	×	×	[AC_X]←[AC_X]+data8 加立即数到累加器A或B中
	ADDD	data16	3	4					×	×	×	×	[D]←[D]+data16 加16位数到双累加器中，高位字节作操作码，低位字节作为高位字节
	ANDA ANDB	data8	2	2					×	×	0		[AC_X]←[AC_X]∧data8 立即数和累加器A或B逻辑"与"
	BITA BITB	data8	2	2					×	×	0		[AC_X]∧data8 立即数和累加器A或B逻辑"与"，但只改变状态寄存器
	CMPA CMPB	data8	2	2			×		×	×	×	×	[AC_X]−data8 从累加器A或B中减去立即数，但只改变状态寄存器
	CMPD	data16	4	5					×	×	×	×	[D]−data16 从双累加器中减去立即数，但只改变状态寄存器

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
立即数操作指令(立即数运算指令)	CMPS CMPU CMPY	data16	4	5					×	×	×	×	〔reg〕－data16 从指定寄存器(S，U，X或Y)中减去立即数，但只影响状态寄存器
	CMPX	data16	3	4					×	×	×	×	
	EORA EORB	data8	2	2					×	×	0		〔AC_X〕←〔AC_X〕∀data8 立即数和累加器A或B内容逻辑"异或"
	ORA ORB	data8	2	2					×	×	0		〔AC_X〕←〔AC_X〕∨data8 立即数和累加器A或B内容逻辑"或"
	SBCA SBCB	data8	2	2			×		×	×	×	×	〔AC_X〕←〔AC_X〕－data－C 从累加器A或B中减去立即数和借位位
	SUBA SUBB	data8	2	2			×		×	×	×	×	〔AC_X〕←〔AC_X〕－data8 从累加器A或B中减去立即数
	SUBD	data16	3	4					×	×	×	×	〔D〕←〔D〕－data16 从双累加器D中减立即数
转移指令	BRA	disp8	2	3									〔PC〕←〔PC〕＋disp8＋2 无条件分支相对转移到程序计数器当前内容所指向的程序上去
	JMP	adr8 adr16 ind, forms	2 3 2＋	3 4 3＋									〔PC〕←EA 无条件转移到使用基本页面直接寻址、直接扩充寻址、变址寻址、间接寻址所找到的绝对有效地址上
	LBRA	disp16	3	5									〔PC〕←〔PC〕＋disp16＋3 无条件长分支相对转移到程序计数器当前内容所指的程序中去
	TFR	R16, PC	2	7									〔PC〕←〔R16〕 无条件转移到指定的16位寄存器(D，S，U，X或Y)内容所指的地址
转子程序和由子程序返回指令	BSR	disp8	2	7									〔(S)－1〕←〔PC(L0)〕　〔(S)－2〕←〔PC(HI)〕 〔S〕←〔S〕－2　〔PC〕←(PC)＋disp8＋2 在执行分支转移前，先把程序计数器当时内容存到硬件堆栈中，再无条件分支转移到程序计数器当前内容相对应的子程序中去。

续表

分类	记忆符	操作数	字节数	周期数	条件码								操作内容
					E	F	H	I	N	Z	V	C	
转子程序和由子程序返回指令	EXG	R16, PC	2	8									〔R16〕←→〔PC〕 程序计数器当时内容存到指定的16位寄存器(D, S, U, X或Y)中，同时寄存器内容传到程序计数器中 (即程序计数器和指定的16位寄存器内容互换)，程序无条件转移到指定的16位寄存器内容所指的地址，这条指令可用来访问子程序和由子程序返回，这个指定的16位寄存器充当一个程序连接器。
	JSR	adr8 adr16 ind, forms	2 3 2+	7 8 7+									〔(S)-1〕←〔PC(L0)〕 〔(S)-2〕←〔PC(H1)〕 (S)←(S)-2 (PC)←EA 执行转移前，把程序计数器当时内容存到硬件堆栈中，再无条件转移到基本页面直接、直接扩展、间接或变址寻址所得到的绝对(有效)地址的子程序上
	LBSR	disp16	3	9									〔(S)-1〕←〔PC(L0)〕 〔(S)-2〕←〔PC(H1)〕 (S)←(S)-2 (PC)←(PC)+disp16+3 执行分支转移前，把程序计数器当时内容存到硬件堆栈中，再无条件长(多位)分支转移到程序计数器现在内容相对应的子程序中
	PULS PULU	PC, reg List	2	5+									从子程序返回并按后缀字节指定的硬件或用户堆栈装载其他寄存器。后缀字节第七位必须是1，以使程序计数器也在被装载的寄存器之列，关于PULS和PULU操作，可参看此指令系统一览表堆栈功能部分
	RTS		1	5									〔PC(H1)〕←〔(S)〕 〔PC(L0)〕←〔(S)+1〕 (S)←(S)+2 从子程序返回：由硬件栈顶弹出程序计数器内容，并且栈指针加2

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
条件分支转移指令													如果结果是真，则
													[PC]←[PC]+disp8+2
	BCC	disp8	2	3									$C=0$
	BCS	disp8	2	3									$C=1$
	BEQ	disp8	2	3									$Z=1$
	BGE	disp8	2	3									$N \veebar V=0$
	BGT	disp8	2	3									$Z \vee (N \veebar V)=0$
	BHI	disp8	2	3									$C \vee Z=0$
	BHS	disp8	2	3									$C=0$
	BLE	disp8	2	3									$Z \vee (N \veebar V)=1$
	BLO	disp8	2	3									$C=1$
	BLS	disp8	2	3									$C \vee Z=1$
	BLT	disp8	2	3									$N \veebar V=1$
	BMI	disp8	2	3									$N=1$
	BNE	disp8	2	3									$Z=0$
	BPL	disp8	2	3									$N=0$
	BVC	disp8	2	3									$V=0$
	BVS	disp8	2	3									$V=1$
													如果结果是假，则[PC]←[PC]+2
													注意，BHS 和 BCC 是对同样操作码不同的记忆符，BLO 和 BCS 亦然
													如果结果是真，则
													[PC]←[PC]+disp16+4
	LBCC	disp16	4	5(6)									$C=0$
	LBCS	disp16	4	5(6)									$C=1$
	LBEQ	disp16	4	5(6)									$Z=1$
	LBGE	disp16	4	5(6)									$N \veebar V=0$
	LBGT	disp16	4	5(6)									$Z \vee (N \veebar V)=0$
	LBHI	disp16	4	5(6)									$C \vee Z=0$
	LBHS	disp16	4	5(6)									$C=0$
	LBLE	disp16	4	5(6)									$Z \vee (N \veebar V)=1$
	LBLO	disp16	4	5(6)									$C=1$

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
条件分支转移指令	LBLS	disp16	4	5(6)									$C \vee Z = 1$
	LBLT	disp16	4	5(6)									$N \forall V = 1$
	LBMI	disp16	4	5(6)									$N = 1$
	LBNE	disp16	4	5(6)									$Z = 0$
	LBPL	disp16	4	5(6)									$N = 0$
	LBVC	disp16	4	5(6)									$V = 0$
	LBVS	disp16	4	5(6)									$V = 1$ 如果所得结果是假，则(PC)←(PC)+4 注意，LBHS 和 LBCC 是同样操作码不同的记忆符，LBLO 和 LBCS 亦然。 如果执行长(多位)分支转移指令，则取6个执行周期,其余取5个执行周期
寄存器间传送指令	EXG	R_1, R_2	2	8									$(R_1) \leftrightarrow (R_2)$ 指定寄存器内容互换 (除 R_1 或 R_2 是状态寄存器外)不影响状态寄存器 CC
	TFR	R_1, R_2	2	7									$(R_2) \leftarrow (R_1)$ R_1 寄存器传送到 R_2 寄存器 (除 R_2 是状态寄存器外)不影响状态寄存器 CC
寄存器间操作指令	ABX		1	3									[X]←[X]+[B] 把累加器B中的无符号数加到变址寄存器X中
	MUL		1	11						×		×	[D]←[A]×[B] 把累加器A和B的无符号数相乘,并把结果放入D,进位位取累加器B中第七位数
	SEX		1	2					×	×			如果累加器B第七位是1,则 [A]←FF_{16} 如果累加器B第七位是0,则 [A]←00_{16} 把B中8位2的补数变转为D中16位2的补数
寄存器操作指令	ASLA ASLB		1	2			×		×	×	×	×	C ← 7 ←— 0 ←0, $V = N \forall C$ (AC_X) 累加器A或B的内容算术左移，第0位置0

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
寄存器操作指令	ASRA ASRB		1	2			×		×	×		×	7→0 (ACx) →C（第七位循环保持） 累加器A或B内容右移，第七位保留不变
	CLRA CLRB		1	2					0	1	0	0	$[AC_X]\leftarrow 00_{16}$ 清除累加器A或B
	COMA COMB		1	2					×	×	0	1	$[AC_X]\leftarrow[\overline{AC_X}]$ 把累加器A或B的内容取反
	DAA		1	2					×	×	×	×	把累加器A十进制调整,变累加器A的数(假定是BCD运算的二进制和)为BCD码。如果进位位预先被置了位,或者调整结果带进位位,则进位位被置位
	DECA DECB		1	2					×	×	×		$[AC_X]\leftarrow[AC_X]-1$ 累加器A或B的内容减1，如果结果是$7F_{16}$，则置位溢出位,其余则清除溢出位
	INCA INCB		1	2					×	×	×		$[AC_X]\leftarrow[AC_X]+1$ 累加器A或B内容加1，如果结果是80_{16}，则置位溢出位,其余则清除溢出位
	LSLA LSLB		1	2			×		×	×	×	×	C←7←0 (ACx)←0，$V=N\forall C$ A或B累加器内容逻辑左移，第0位被置0，同ACL
	LSRA LSRB		1	2					0	×		×	0→7→0 (ACx)→C A或B累加器内容逻辑右移，第七位置0
	NEGA NEGB		1	2			×		×	×	×	×	$[AC_X]\leftarrow 00_{16}-[AC_X]$ 把累加器A或B的内容求2的补码。如果结果是00_{16}，置位进位位，其他则清除进位位。如果结果是80_{16}，置位溢出位，其他则清除溢出位。

续表

分类	记忆符	操作数	字节数	周期数	E	F	H	I	N	Z	V	C	操作内容
寄存器操作指令	ROLA ROLB		1	2					×	×	×	×	C ← [7 ←— 0] ← (循环), [AC_X], V = N⊻C 累加器A或B内容带进位位循环左移
	RORA RORB		1	2					×	×			C → [7 —→ 0] (循环), [AC_X] 累加器A或B内容带进位位循环右移
	TSTA TSTB		1	2					×	×		×	[AC_X] − 00_{16} 测试累加器A或B中的数是0或是负值否?
	LEAX LEAY LEAS LEAU	ind forms ind forms	2+ 2+	 4+						×			[reg]←EA 按照变址/间址寻址方式中任一种形成有效地址，并把这有效地址装载到指定的寄存器(X、Y、S或U)中，LEA指令主要被用来计算一个有效的重复使用的立即地址，但也可用来执行16位算术运算。
堆栈指令	PSHS PSHU	reg list	2	5+									测试状态位字节内容，并按下列条件把寄存器内容存到指定的堆栈中 $b_7=1$; [SP]←[SP]−1, [(SP)]←[PC(L0)] [SP]←[SP]−1, [(SP)]←[PC(H1)] $b_6=1$; [SP]←[SP]−1, [(SP)]←[U(L0)]/[S(L0)] [SP]←[SP]−1, [(SP)]←[U(H1)]/[S(H1)] $b_5=1$; [SP]←[SP]−1, [(SP)]←[Y(L0)] [SP]←[SP]−1, [(SP)]←[Y(H1)] $b_4=1$; [SP]←[SP]−1, [(SP)]←[X(L0)] [SP]←[SP]−1, [(SP)]←[X(H1)] $b_3=1$; (SP)←(SP)−1, [(SP)]←[DP]

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
堆													$b_2=1$; (SP)←(SP)－1, [(SP)]←(B) $b_1=1$; (SP)←(SP)－1, [(SP)]←(A) $b_0=1$; (SP)←(SP)－1, [(SP)]←(CC) 除去该堆栈指示器外，一个、部分或全部寄存器 的内容 压入指定的堆栈中。每压入一个字节，执行时间增加一个机器的周期。
栈指令	PULS PULU	reg list	2	5＋									测试状态位 字节 的内容并按下列状态把指定堆栈的内容装载到指定的寄存器中 $b_0=1$; [CC]←[(SP)], [SP]←[SP]＋1 $b_1=1$; [A]←[(SP)], [SP]←[SP]＋1 $b_2=1$, [B]←[(SP)], [SP]←[SP]＋1 $b_3=1$, [DP]←[(SP)], [SP]←[SP]＋1 $b_4=1$; [X(HI)]←[(SP)], [SP]←[SP]＋1 [X(LO)]←[(SP)], [SP]←[SP]＋1 $b_5=1$; [Y(HI)]←[(SP)], [SP]←[SP]＋1 [Y(LO)]←[(SP)], [SP]←[SP]＋1 $b_6=1$; [U(HI)]/[S(HI)]←[(SP)], [SP]←[SP]＋1 [U(LO)]/[S(LO)]←[(SP)], [SP]←[SP]＋1 $b_7=1$; [PC(HI)]←[(SP)], [SP]←[SP]＋1

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
堆栈指令													〔PC(L0)〕←〔(SP)〕，〔SP〕←〔SP〕+1 从指定的堆栈中弹出一个、几个或全部寄存器的内容，(除去堆栈指示器外)，状态寄存器的各位由堆栈中弹回的字节决定。
中断指令	CWAI	data8	2	20									〔CC〕←〔CC〕∧data8，这可以清除状态位 CC E←1 〔S〕←〔S〕-1，〔(S)〕←〔PC(L0)〕 〔S〕←〔S〕-1，〔(S)〕←〔PC(Hi)〕 〔S〕←〔S〕-1，〔(S)〕←〔U(L0)〕 〔S〕←〔S〕-1，〔(S)〕←〔U(Hi)〕 〔S〕←〔S〕-1，〔(S)〕←〔Y(L0)〕 〔S〕←〔S〕-1，〔(S)〕←〔Y(Hi)〕 〔S〕←〔S〕-1，〔(S)〕←〔X(L0)〕 〔S〕←〔S〕-1，〔(S)〕←〔X(Hi)〕 〔S〕←〔S〕-1，〔(S)〕←〔DP〕 〔S〕←〔S〕-1，〔(S)〕←〔B〕 〔S〕←〔S〕-1，〔(S)〕←〔A〕 〔S〕←〔S〕-1，〔(S)〕←〔CC〕 全部寄存器内容压入硬堆栈并等待中断。当非屏蔽中断来到时，转相应的中断服务程序。注意：将进入快速中断服务程序时的全部寄存器内容存栈保留。而由于CWAI设置了E标志位，RTI指令将正确地恢复这些寄存器的内容。CWAI不悬空系统总线。
	RTI		1	6/15									按照状态寄存器中E标志位的值的情况，把硬堆栈中内容弹回到寄存器中。若E=0，只弹回部分寄存器内容 〔CC〕←〔(S)〕，〔S〕←〔S〕+1 〔PC(Hi)〕←〔(S)〕，〔S〕←〔S〕+1 〔PC(L0)〕←〔(S)〕，〔S〕←〔S〕+1 如果E=1，弹回全部寄存器内容 〔CC〕←〔(S)〕，〔S〕←〔S〕+1 〔A〕←〔(S)〕，〔S〕←〔S〕+1 〔B〕←〔(S)〕，〔S〕←〔S〕+1 〔DP〕←〔(S)〕，〔S〕←〔S〕+1

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
中断指令													〔X(HI)〕←〔(S)〕，〔S〕←〔S〕+1 〔X(LO)〕←〔(S)〕，〔S〕←〔S〕+1 〔Y(HI)〕←〔(S)〕，〔S〕←〔S〕+1 〔Y(LO)〕←〔(S)〕，〔S〕←〔S〕+1 〔U(HI)〕←〔(S)〕，〔S〕←〔S〕+1 〔U(LO)〕←〔(S)〕，〔S〕←〔S〕+1 〔PC(HI)〕←〔(S)〕，〔S〕←〔S〕+1 〔PC(LO)〕←〔(S)〕，〔S〕←〔S〕+1 状态寄存器各位同样从硬堆栈中弹出
	SWI SWI2 SWI3		1 2 2	19 20 20	1 1 1	1							全部寄存器内容存硬堆栈，控制转中断子程序，中断向量地址如下排列 FFFA和FFFB为SWI用 FFF4和FFF5为SWI2用 FFF2和FFF3为SWI3用 E←1 〔S〕←〔S〕-1，〔(S)〕←〔PC(LO)〕 〔S〕←〔S〕-1，〔(S)〕←〔PC(HI)〕 〔S〕←〔S〕-1，〔(S)〕←〔U(LO)〕 〔S〕←〔S〕-1，〔(S)〕←〔U(HI)〕 〔S〕←〔S〕-1，〔(S)〕←〔Y(LO)〕 〔S〕←〔S〕-1，〔(S)〕←〔Y(HI)〕 〔S〕←〔S〕-1，〔(S)〕←〔X(LO)〕 〔S〕←〔S〕-1，〔(S)〕←〔X(HI)〕 〔S〕←〔S〕-1，〔(S)〕←〔DP〕 〔S〕←〔S〕-1，〔(S)〕←〔B〕 〔S〕←〔S〕-1，〔(S)〕←〔A〕 〔S〕←〔S〕-1，〔(S)〕←〔CC〕 〔PC〕←中断向量 注意：SWI不适于经常性和快速的中断使用，而SWI2和SWI3不受限制

续表

分类	记忆符	操作数	字节数	周期数	条件码 E	F	H	I	N	Z	V	C	操作内容
中断指令	SYNC		1	2									停止处理指令，挂起系统总线并等待中断。当中断发生时，按下列情况继续处理 a. 如果允许中断，控制转到服务程序。 b. 如果不允许中断，继续执行下条指令。
状态寄存器指令	ANDCC	data8	2	3									[CC]←[CC]∧data8 立即数和状态寄存器内容逻辑"与"，用 0 和状态寄存器中某些位相"与"以清除状态寄存器中某些位。
	ORCC	data8	2	3									[CC]←[CC]∨data8 立即数和状态寄存器内容逻辑"或"，用 1 和状态寄存器中某些位逻辑或以设置该位
空操作	BRN	disp8	2	3									不转移，这是一个空操作
	LBRN	disp16	5	4									不长转移，这是一个空操作
	NOP		2	1									空操作

附录 6　6809变址和间址型综合一览表

分类	形式	非间址 汇编程序类型	非间址 后缀字节操作码	周期数	字节数	间址 汇编程序类型	间址 后缀字节操作码	周期数	字节数
由寄存器中偏移常数值	不偏移	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 位偏移	n,R	0RRnnnnn	1	0		没有8位		
	8 位偏移	nn,R	1RR01000	1	1	[nn,R]	1RR11000	4	1
	16位偏移	mmnn,R	1RR01001	4	2	[mmnn,R]	1RR11001	7	2
由寄存器中偏移累加器中值	A-寄存器偏移	A,R	1RR00110	1	0	[A,R]	1RR10110	4	0
	B-寄存器偏移	B,R	1RR00101	1	0	[B,R]	1RR10101	4	0
	D-寄存器偏移	D,R	1RR01011	4	0	[D,R]	1RR11011	7	0
寄存器增/减	增 1	,R +	1RR00000	2	0		不允许		
	增 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	减 1	,− R	1RR00010	2	0		不允许		
	减 2	,− R	1RR00011	3	0	[,− − R]	1RR10011	6	0
由 PC 中偏移常数值	8 位偏移	label,PCR	1XX01100	1	1	[label,PCR]	1XX11100	4	1
	16位偏移	label,PCR	1XX01101	5	2	[label,PCR]	1XX11001	8	2
扩展，间接寻址	16位地址		—	—	—	[mmnn]	10011111	5	2

R = X, Y, U, 或 S　　RR: 00 = X　　10 = U

XX = 任意　　　　　　01 = Y　　11 = S

注意：这张表格与 Motorola 公司命名法一致，用方括号表示汇编程序寻址方式是间址型。因此，此处的[ ]与附录 E 中的用法不同。

附录7　6809指令码、字节数和执行时间

地址类型 →	隐	含		立	即		直	接		扩	展		变址/间址			相	对		注
操作类型 →				data8 或 data16			adr8			adr16			参看附录6			标号或偏移量			
指令符号 ↓	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	
ABX	3A	3	1																
ADCA				89	2	2	99	4	2	B9	5	3	A9	4+	2+				
ADCB				C9	2	2	D9	4	2	F9	5	3	E9	4+	2+				
ADDA				8B	2	2	9B	4	2	BB	5	3	AB	4+	2+				
ADDB				CB	2	2	DB	4	2	FB	5	3	EB	4+	2+				
ADDD				C3	4	3	D3	6	2	F3	7	3	E3	6+	2+				
ANDA				84	2	2	94	4	2	B4	5	3	A4	4+	2+				
ANDB				C4	2	2	D4	4	2	F4	5	3	E4	4+	2+				
ANDCC				1C	3	2													
ASL							08	6	2	78	7	3	68	6+	2+				
ASLA	4B	2	1																
ASLB	5B	2	1																
ASR							07	6	2	77	7	3	67	6+	2+				
ASRA	47	2	1																
ASRB	57	2	1																
BCC																24	3	2	
BCS																25	3	2	
BEQ																27	3	2	
BGE																2C	3	2	
BGT																2E	3	2	
BHI																22	3	2	
BHS																24	3	2	
BITA				85	2	2	95	4	2	B5	5	3	A5	4+	2+				
BITB				C5	2	2	D5	4	2	F5	5	3	E5	4+	2+				
BLE																2F	3	2	
BLO																25	3	2	
BLS																23	3	2	
BLT																2D	3	2	
BMI																2B	3	2	
BNE																26	3	2	

续表

地址类型 →	隐	含		立	即		直	接		扩	展		变址/间址			相	对		注
操作类型 →				data8 或 data16			adr8			adr16			参看附录 6			标号或偏移量			
指令符号 ↓	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	
BPL																2A	3	2	
BRA																20	3	2	
BRN																21	3	2	
BSR																8D	3	2	
BVC																28	3	2	
BVS																29	3	2	
CLR							0F	6	2	7F	7	3	6F	6+	2+				
CLRA	4F	2	1																
CLRB	5F	2	1																
CMPA				81	2	2	91	4	2	B1	5	3	A1	4+	2+				
CMPB				C1	2	2	D1	4	2	F1	5	3	E1	4+	2+				
CMPC				1083	5	4	1093	7	3	10B3	8	4	10A3	7+	3+				
CMPS				11BC	5	4	119C	7	3	11BC	8	4	11AC	7+	3+				
CMPU				1183	5	4	1193	7	3	11B3	8	4	11A3	7+	3+				
CMPX				8C	4	3	9C	6	2	BC	7	3	AC	6+	2+				
CMPY				108C	5	4	109C	7	3	10BC	8	4	10AC	7+	3+				
COM							03	6	2	73	7	3	63	6+	2+				
COMA	43	2	1																
COMB	53	2	1																
CWAI				3C	20	2													
DAA	19	2	1																
DEC							0A	6	2	7A	7	3	6A	6+	2+				
DECA	4A	2	1																
DECB	5A	2	1																
EORA				8B	2	2	98	4	2	B8	5	3	A8	4+	2+				
EORB				CB	2	2	D8	4	2	F8	5	3	E8	4+	2+				
EXG				1E	8	2													
INC							0C	6	2	7C	7	3	6C	6+	2+				
INCA	4C	2	1																
INCB	5C	2	1																

续表

地址类型 →	隐含			立即			直接			扩展			变址/间址			相对			注
操作类型 →				data8 或 data16			adr8			adr16			参看附录 6			标号或偏移量			
指令符号 ↓	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	
JMP							0E	3	2	7E	4	3	6E	3+	2+				
JSR							9D	7	2	BD	8	3	AD	7+	2+				
LBCC																1024	5(6)	4	1
LPCB																1025	5(6)	4	1
LBEO																1027	5(6)	4	1
LBGB																102C	5(6)	4	1
LBGT																102E	5(6)	4	1
LBHI																1022	5(6)	4	1
LBHS																1024	5(6)	4	1
LBLE																102F	5(6)	4	1
LBLO																1025	5(6)	4	1
LBLS																1023	5(6)	4	1
LBLT																102D	5(6)	4	1
LBM																102B	5(6)	4	1
LBME																1026	5(6)	4	1
LBPL																102A	5(6)	4	1
LBRA																16	5	3	
LBRN																1021	5	4	
LBSR																17	9	3	
LBVC																1028	5(6)	4	1
LBVS																1029	5(6)	4	1
LDA				86	2	2	86	4	2	B6	5	3	A6	4+	2+				
LDB				C6	2	2	C6	4	2	F6	5	3	E6	4+	2+				
LDD				CC	3	3	CC	5	2	FC	6	3	EC	5+	2+				
LDS				10CE	4	4	10DE	6	3	10FE	7	4	10EE	6+	3+				
LDU				CE	3	3	DE	5	2	FE	6	3	EE	5+	2+				
LDX				8E	3	3	9E	5	2	BE	6	3	AE	5+	2+				
LDY				108E	4	4	109E	6	3	10BE	7	4	10AE	6+	3+				
LEAS													32	4+	2+				
LEAU													33	4+	2+				

续表

地址类型 →	隐含			立即			直接			扩展			变址/间址			相对			注
操作类型 →				data8 或 data16			adr8			adr16			参看附录 6			标号或偏移量			
指令符号 ↓	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	
LEAX													30	4 +	2 +				
LEAY													31	4 +	2 +				
LSL							08	6	2	78	7	3	68	6 +	2 +				
LSLA	48	2	1																
LSLB	58	2	1																
LSR							04	6	2	74	7	3	64	6 +	2 +				
LSRA	44	2	1																
LSRB	54	2	1																
MUL	3D	11	1																
NEG							00	6	2	70	7	3	60	6 +	2 +				
NEGA	40	2	1																
NEGB	50	2	1																
NOP	12	2	1																
ORA				8A	2	2	9A	4	2	BA	5	3	AA	4 +	2 +				
ORB				CA	2	2	DA	4	2	FA	5	3	EA	4 +	2				
ORCC				1A	3	2													
PSHS				34	5 +	2													2.3
PSHU				36	5 +	2													2.3
PULS				35	5 +	2													2.3
PULU				37	5 +	2													2.3
ROL							09	6	2	79	7	3	69	6 +	2 +				
ROLA	49	2	1																
ROLB	59	2	1																
ROR							06	6	2	76	7	3	66	6 +	2 +				
RORA	46	2	1																
RORB	56	2	1																
RTI	3B	6/15	1																
RTS	39	5	1																
SBCA				82	2	2	92	4	2	B2	5	3	A2	4 +	2 +				
SBCB				C2	2	2	D2	4	2	F2	5	3	E2	4 +	2 +				

续表

地址类型 →	隐	含		立	即		直	接		扩	展		变址/间址			相	对		注
操作类型 →				data8 或 data16			adr8			adr16			参看附录 6			标号或偏移量			
指令符号 ↓	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	目标码	周期数	字节数	
SEX	1D	2	1																
STA							97	4	2	B7	5	3	A7	4+	2+				
STB							D7	4	2	F7	5	3	E7	4+	2+				
STD							DD	5	2	FD	6	3	ED	5+	2+				
STS							10DF	6	3	10FF	7	4	10EF	6+	3+				
STU							DF	5	2	FF	6	3	EF	5+	2+				
STX							9F	5	2	BF	5	3	AF	5+	2+				
STY							109F	6	3	10BF	7	4	10AF	6+	3+				
SUBA				80	2	2	90	4	2	B0	5	3	A0	4+	2+				
SUBB				C0	2	2	D0	4	2	F0	5	3	E0	4+	2+				
SUBD				83	4	3	93	6	2	B3	7	3	A3	6+	2+				
SWI	3F	19	1																
SWI2	103F	20	2																
SWI3	133F	20	2																
SYNC	13	2	1																
TFR				1F	7	2													
TST							0D	6	2	7D	7	3	6D	6+	2+				2
TSTA	4D	2	1																
TSTB	5D	2	1																

注：1. 若是分支转移，取括弧中的周期数。

2. 这种指令的目标码的立即数总是指规定的编码寄存器内容。

3. 对于压入或弹出指令的每一个字节，一条 PSH 或 PUL 指令，要求 5 个周期加 1 个周期。

附录8　6809指令目标码数字顺序

附录8中出现下列符号和缩写词

adr8	8位地址
adr16	16位地址
data8	8位数据
data16	16位数据
dd	8位数据
dd　dd	16位数据
label	转移或分支跳转的目的地
mm	8位偏移目标码
mm　nn	16位偏移目标码
pp	变址和间址中的后缀字节
qq	8位地址
ssqq	16位地址

6809指令目标码数字顺序

Object Code[1](目标码)	Instruction[2,3](指令)	Addressing Mode(寻址类型)
00 qq	NEG adr8	Base page (direct)(基本页面(直接))
03 qq	COM adr8	Base page (direct)
04 qq	LSR adr8	Base page (direct)
06 qq	ROR adr8	Base page (direct)
07 qq	ASR adr8	Base page (direct)
08 qq	ASL adr8/LSL adr8	Base page (direct)
09 qq	ROL adr8	Base page (direct)
0A qq	DEC adr8	Base page (direct)
0C qq	INC adr8	Base page (direct)
0D qq	TST adr8	Base page (direct)
0E qq	JMP adr8	Base page (direct)
0F qq	CLR adr8	Base page (direct)
1021 mm nn	LBRN label	Relative (相对)
1022 mm nn	LBHI label	Relative
1023 mm nn	LBLS label	Relative
1024 mm nn	LBHS label/LBCC label	Relative
1025 mm nn	LBLO label/LBCS label	Relative
1026 mm nn	LBNE label	Relative
1027 mm nn	LBEQ label	Relative
1028 mm nn	LBVC label	Relative
1029 mm nn	LBVS label	Relative
102A mm nn	KBPL label	Relative
102B mm nn	LBMI label	Relative
102C mm nn	LBGE label	Relative
102D mm nn	LBLT label	Relative
102E mm nn	LBGT label	Relative
102F mm nn	LBLE label	Relative
103F	SWI2	Inherent (隐含)
1083 dd dd	CMPD data16	Immediate (立即)
108C dd dd	CMPY data16	Immediate
108E dd dd	LDY data16	Immediate
1093 qq	CMPD adr8	Base page (direct)

续表

Object Code[1](目标码)	Instruction[2,3](指令)	Addressing Mode(寻址类型)
109C qq	CMPY adr8	Base page (direct)
109E qq	LDY adr8	Base page (direct)
109F qq	STY adr8	Base page (direct)
10A3 ppl	CMPD indexed forms(变址型)	Indexed/indirect(变址/间址)
10AC ppl	CMPY indexed forms	Indexed/indirect
10AE ppl	LDY indexed forms	Indexed/indirect
10AF ppl	STY indexed forms	Indexed/indirect
10B3 ss qq	CMPD adr16	Extended (direct)〔扩展(直接)〕
10BC ss qq	CMPY adr16	Extended (direct)
10BE ss qq	LDY adr16	Extended (direct)
10BF ss qq	STY adr16	Extended (direct)
10CE dd dd	LDS data16	Immediate(立即)
10DE qq	LDS adr8	Base page(direct)
10DF qq	STS adr8	Base page(direct)
10EE ppl	LDS indexed forms	Indexed/indirect
10EF ppl	STS indexed forms	Indexed/indirect
10FE ss qq	LDS adr16	Extended (direct)
10FF ss qq	STS adr16	Extended (direct)
113F	SW13	Inherent
1183 dd dd	CMPU date 16	Immediate
118C dd dd	CMPS data 16	Immediate
1193 qq	CMPU adr8	Base page(direct)
119C qq	CMPS adr8	Base page(direct)
1143 ppl	CMPU indexed forms	Indexed/indirect
11AC ppl	CMPS indexed forms	Indexed/indirect
11B3 ss qq	CMPU adr 16	Extended (direct)
11Bc ss qq	CMPS adr 16	Extended (direct)
12	NOP	Inherent
13	SYNC	Inherent
16mm nn	LBRA label	Relative(相对)
17mm nn	LBSR label	Relative
19	DAA	Inherent
1A dd	ORCC data 8	Immediate
1C dd	ANDCC data 8	Immediate
1D	SEX	Inherent
1E dd	EXG data 8	Register4(寄存器)

续表

Object Code1(目标码)	Instruction2.3(指令)	Addressing Mode(寻址类型)
1F dd	TFR data 8	Register4
20mm	BRA label	Relative
21mm	BRN label	Relative
22mm	BHI label	Relative
23mm	BLS label	Relative
24mm	BCC label/BHS label	Relative
25mm	BCS label/BLO label	Relative
26mm	BNE label	Relative
27mm	BEQ label	Relative
28mm	BVC label	Relative
29mm	BVS label	Relative
2Amm	BPL label	Relative
2Bmm	BMI label	Relative
2Cmm	BGE label	Relative
2Dmm	BLT label	Relative
2Emm	BGT label	Relative
2Fmm	LBE label	Relative
30 ppl	LEAX indexed forms	Indexed/indirect
31 ppl	LEAY indexed forms	Indexed/indirect
32 ppl	LEAS indexed forms	Indexed/indirect
33 ppl	LEAU indexed forms	Indexed/indirect
34 dd	PSHS data8	Register5
35 dd	PULS data8	Register5
36 dd	PSHU data8	Register5
37 dd	PULU data8	Register5
39	RTS	Inherent (Stack)隐含(栈)
3A	ABX	Inherent
3B	RTI	Inherent (Stack)
3C dd	CWAI data8	Immediate
3D	MUL	Inherent
3F	SWI	Inherent
40	NEGA	Accumulator(累加器)
43	COMA	Accumulator
44	LSRA	Accumulator
46	RORA	Accumulator
47	ASRA	Accumulator
48	ASLA/LSLA	Accumulator
49	ROLA	Accumulator
4A	DECA	Accumulator
4C	INCA	Accumulator
4D	TSTA	Accumulator
4F	CLRA	Accumulator
50	NEGB	Accumulator
53	COMB	Accumulator
54	LSRB	Accumulator
56	RORB	Accumulator
57	ASRB	Accumulator
58	ASLB/LSLB	Accumulator

Object Code1(目标码)	Instruction2.3(指令)	Addressing Mode(寻址类型)
59	ROLB	Accumulator
5A	DECB	Accumulator
5C	INCB	Accumulator
5D	TSTB	Accumulator
5F	CLRB	Accumulator
60 ppl	NEG indexed forms	Indexed/indirect
63 ppl	COM indexed forms	Indexed/indirect
64 ppl	LSR indexed forms	Indexed/indirect
66 ppl	ROR indexed forms	Indexed/indirect
67 ppl	ASR indexed forms	Indexed/indirect
68 ppl	ASL/LSL indexed forms	Indexed/indirect
69 ppl	ROL indexed forms	Indexed/indirect
6A ppl	DEC indexed forms	Indexed/indirect
6C ppl	INC indexed forms	Indexed/indirect
6D ppl	TST indexed forms	Indexed/indirect
6E ppl	JMP indexed forms	Indexed/indirect
6F ppl	CLR indexed forms	Indexed/indirect
70 ss qq	NEG adr 16	Extended (direct)
73 ss qq	COM adr 16	Extended (direct)
74 ss qq	LSR adr 16	Extended (direct)
76 ss qq	ROR adr 16	Extended (direct)
77 ss qq	ASR adr 16	Extended (direct)
78 ss qq	ASL adr16/LSL adr16	Extended (direct)
79 ss qq	ROL adr 16	Extended (direct)
7A ss qq	DEC adr 16	Extended (direct)
7C ss qq	INC adr 16	Extended (direct)
7D ss qq	TST adr 16	Extended (direct)
7E ss qq	JMP adr 16	Extended (direct)
7F ss qq	CLR adr 16	Extended (direct)
80 dd	SUBA data 8	Immediate
81 dd	CMPA data 8	Immediate
82 dd	SBCA data 8	Immediate
83 dd dd	SUBD data 16	Immediate
84 dd	ANDA data 8	Immediate
85 dd	BITA data 8	Immediate
86 dd	LDA data 8	Immediate
88 dd	EORA data88	Immediate
89 dd	ADCA data 8	Immediate
8A dd	ORA data 8	Immediate
8B dd	ADDA data 8	Immediate
8C dd dd	CMPX data 16	Immediate
8D mm	BSR label	Relative
8E dd dd	LDX data 16	Immediate
90 qq	SUBA adr 8	Base page (direct)
91 qq	CMPA adr 8	Base page (direct)
92 qq	SBCA adr 8	Base page (direct)
93 qq	SUBD adr 8	Base page (direct)
94 qq	ANDA adr 8	Base page (direct)

Object Code1(目标码)	Instruction2.3(指令)	Addressing Mode(寻址类型)
95 qq	BITA adr8	Base page (direct)
96 qq	LDA adr8	Base page (direct)
97 qq	STA adr8	Base page (direct)
98 qq	EORA adr8	Base page (direct)
99 qq	ADCA adr8	Base page (direct)
9A qq	ORA adr8	Base page (direct)
9B qq	ADDA adr8	Base page (direct)
9C qq	CMPX adr8	Base page (direct)
9D qq	JSR adr8	Base page (direct)
9E qq	LDX adr8	Base page (direct)
9F qq	STX adr8	Base page (direct)
A0 ppl	SUBA indexed forms(变址型)	Indexed/indirect
A1 ppl	CMPA indexed forms	Indexed/indirect
A2 ppl	SBCA indexed forms	Indexed/indirect
A3 ppl	SUBD indexed forms	Indexed/indirect
A4 ppl	ANDA indexed forms	Indexed/indirect
A5 ppl	BITA indexed forms	Indexed/indirect
A6 ppl	LDA indexed forms	Indexed/indirect
A7 ppl	STA indexed forms	Indexed/indirect
A8 ppl	EORA indexed forms	Indexed/indirect
A9 ppl	ADCA indexed forms	Indexed/indirect
AA ppl	ORA indexed forms	Indexed/indirect
AB ppl	ADDA indexed forms	Indexed/indirect
AC ppl	CMPX indexed forms	Indexed/indirect
AD ppl	JSR indexed forms	Indexed/indirect
AE ppl	LDX indexed forms	Indexed/indirect
AF ppl	STX indexed forms	Indexed/indirect
B0 ss qq	SUBA adr16	Extended (direct)
B1 ss qq	CMPA adr16	Extended(direct)
B2 ss qq	SBCA adr16	Extended(direct)
B3 ss qq	SUBD adr16	Extended(direct)
B4 ss qq	ANDA adr16	Extended(direct)
B5 ss qq	BITA adr16	Extended(direct)
B6 ss qq	LDA adr16	Extended(direct)
B7 ss qq	STA adr16	Extended(direct)
B8 ss qq	EORA adr16	Extended(direct)
B9 ss qq	ADCA adr16	Extended(direct)
BA ss qq	ORA adr16	Extended(direct)
BB ss qq	ADDA adr16	Extended(direct)
BC ss qq	CMPX adr16	Extended(direct)
BD ss qq	JSR adr16	Extended(direct)
BE ss qq	LDX adr16	Extended(direct)
BF ss qq	STX adr16	Extended(direct)
C0 dd	SUBB data8	Immediate
C1 dd	CMPB data8	Immediate
C2 dd	SBCB data8	Immediate
C3 dd dd	ADDD data16	Immediate
C4 dd	ANDB data8	Immediate

Object Code1(目标码)	Instruction2.3(指令)	Addressing Mode(寻址类型)
C5 dd	BITB data8	Immediate
C6 dd	LDB data8	Immediate
C8 dd	EORB data8	Immediate
C9 dd	ADCB data8	Immediate
CA dd	ORB data8	Immediate
CB dd	ADDB data8	Immediate
CC dd dd	LDD data16	Immediate
CE dd dd	LDU data16	Immediate
D0 qq	SUBB adr8	Base page (direct)
D1 qq	CMPB adr8	Base page (direct)
D2 qq	SBCB adr8	Base page (direct)
D3 qq	ADDD adr8	Base page (direct)
D4 qq	ANDB adr8	Base page (direct)
D5 qq	BITB adr8	Base page (direct)
D6 qq	LDB adr8	Base page (direct)
D7 qq	STB adr8	Base page (direct)
D8 qq	EORB adr8	Base page (direct)
D9 qq	ADCB adr8	Base page (direct)
DA qq	ORB adr8	Base page (direct)
DB qq	ADDB adr8	Base page (direct)
DC qq	LDD adr8	Base ppge (direct)
DD qq	STD adr8	Base aage (direct)
DE qq	LDU adr8	Base page (direct)
DF qq	STU adr8	Base page (direct)
E0 ppl	SUBB indexed forms	Indexed/indirect
E1 ppl	CMPB indexed forms	Indexed/indirect
E2 ppl	SBCB indexed forms	Indexed/indirect
E3 ppl	ADDD indexed forms	Indexed/indirect
E4 ppl	ANDB indexed forms	Indexed/indirect
E5 ppl	BITB indexed forms	Indexed/indirect
E6 ppl	LDB indexed forms	Indexed/indirect
E7 ppl	STB indexed forms	Indexed/indirect
E8 ppl	EORB indexed forms	Indexed/indirect
E9 ppl	ADCB indexed forms	Indexed/indirect
EA ppl	ORB indexed forms	Indexed/indirect
EB ppl	ADDB indexed forms	Indexed/indirect
EC ppl	LDD indexed forms	Indexed/indirect
ED ppl	STD indexed forms	Indexed/indirect
EE ppl	LDU indexed forms	Indexed/indirect
EF ppl	STU indexed forms	Indexed/indirect
F0 ss qq	SUBB adr16	Extended(direct)
F1 ss qq	CMPB adr16	Extended(direct)
F2 ss qq	SBCB adr16	Extended(direct)
F3 ss qq	ADDD adr16	Extended(direct)
F4 ss qq	ANDB adr16	Extended(direct)
F5 ss qq	BITB adr16	Extended(direct)
F6 ss qq	LDB adr16	Extended(direct)
F7 ss qq	STB adr16	Extended(direct)

Object Code1(目标码)	Instruction2.3(指令)	Addressing Mode(寻址类型)
F8 ss qq	EORB adr16	Extended(direct)
F9 ss qq	ADCB adr16	Extended(direct)
FA ss qq	ORB adr16	Extended(direct)
FB ss qq	ADDB adr16	Extended(direct)
FC ss qq	LDD adr16	Extended(direct)
FD ss pp	STD adr16	Extended(direct)
FE ss pp	LDU adr16	Extended(direct)
FF ss pp	STU adr16	Extended(direct)

注：

1. 后缀字节可后跟2个字节、1个字节或不跟字节。更详细的可参看附录6和第三章中的后缀字节的论述。附录9列出了所有可能的后缀字节和它的操作形式。
2. 有些指令有两个记忆符，这时，我们用一斜线（/）把两者隔开表示。
3. 附录6表示出在变址和间址寻址方式中操作数的变址形式。
4. 在EXG和TFR指令中，处理器将把第二个字节（立即数）作为指定的源、目寄存器。
5. 在PSHS，PULS，PSHU，PULU指令中，处理器将把第二个字节（立即数）的内容作为进出堆栈的各个寄存器。

附录9　6809后缀字节数字顺序

后缀字节	操作类型	后缀字节	操作类型	后缀字节	操作类型	后缀字节	操作类型
00	0,X	21	1,Y	42	2,U	63	3,S
01	1,X	22	2,Y	43	3,U	64	4,S
02	2,X	23	3,Y	44	4,U	65	5,S
03	3,X	24	4,Y	45	5,U	66	6,S
04	4,X	25	5,Y	46	6,U	67	7,S
05	5,X	26	6,Y	47	7,U	68	8,S
06	6,X	27	7,Y	48	8,U	69	9,S
07	7,X	28	8,Y	49	9,U	6A	10,S
08	8,X	29	9,Y	4A	10,U	6B	11,S
09	9,X	2A	10,Y	4B	11,U	6C	12,S
0A	10,X	2B	11,Y	4C	12,U	6D	13,S
0B	11,X	2C	12,Y	4D	13,U	6E	14,S
0C	12,X	2D	13,Y	4F	14,U	6F	15,S
0D	13,X	2E	14,Y	4F	15,U	70	-16,S
0E	14,X	2F	15,Y	50	-16,U	71	-15,S
0F	15,X	30	-16,Y	51	-15,U	72	-14,S
10	-16,X	31	-15,Y	52	-14,U	73	-13,S
11	-15,X	32	-14,Y	53	-13,U	74	-12,S
12	-14,X	33	-13,Y	54	-12,U	75	-11,S
13	-13,X	34	-12,Y	55	-11,U	76	-10,S
14	-12,X	35	-11,Y	56	-10,U	77	-9,S
15	-11,X	36	-10,Y	57	-9,U	78	-8,S
16	-10,X	37	-9,Y	58	-8,U	79	-7,S
17	-9,X	38	-8,Y	59	-7,U	7A	-6,S
18	-8,X	39	-7,Y	5A	-6,U	7B	-5,S
19	-7,X	3A	-6,Y	5B	-5,U	7C	-4,S
1A	-6,X	3B	-5,Y	5C	-4,U	7D	-3,S
1B	-5,X	3C	-4,Y	5D	-3,U	7E	-2,S
1C	-4,X	3D	-3,Y	5E	-2,U	7F	-1,S
1D	-3,X	3E	-2,Y	5F	-1,U	80	,X+
1E	-2,X	3F	-1,Y	60	0,S	81	,X++
1F	-1,X	40	0,U	61	1,S	82	,-X
20	0,Y	41	1,U	62	2,S	83	,--X
84	,X	A3	,--Y	C2	,-U	E1	,S++
85	B,X	A4	,Y	C3	,--U	E2	,-S
86	A,X	A5	B,Y	C4	,U	E3	,--S
88	nn,X	A6	A,Y	C5	B,U	E4	,S
89	mmnn,X	A8	nn,Y	C6	A,U	E5	B,S
8B	D,X	A9	mmnn,Y	C8	nn,U	E6	A,S
8C	nn,PC[2]	AB	D,Y	C9	mmnn,U	E8	nn,S
8D	mmnn,PC[2]	AC	nn,PC[2]	CB	D,U	E9	mmnn,S
91	(,X++)	AD	mmnn,PC[2]	CC	nn,PC[2]	EB	D,S
93	(--X)	B1	(,Y++)	CD	mmnn,PC[2]	EC	nn,PC[2]
94	(,X)	B3	(,--Y)	D1	(,U++)	ED	mmnn,PC[2]
95	(B,X)	B4	(,Y)	D3	(,--U)	F1	(S++)

续表

后缀字节	操作类型	后缀字节	操作类型	后缀字节	操作类型	后缀字节	操作类型
96	(A,X)	B5	(B,Y)	D4	(,U)	F3	(,--S)
98	(nn,X)	B6	(A,Y)	D5	(B,U)	F4	(,S)
99	(mmnn,X)	B8	(nn,Y)	D6	(A,U	F5	(B,S)
9B	(D,X)	B9	(mmmm,Y)	D8	(nn,U)	F6	(A,S)
9C	(nn,PC)[3]	BB	(D,Y)	D9	(mmnn,U)	F8	(nn,S)
9D	(mmnn,PC)[3]	BC	(nn,PC)[3]	DB	(D,U)	F9	(mmnn,S)
9E	(mmnn)	BD	(mmnn,PC)[3]	DC	(nn,PC)[3]	FB	(D,S)
A0	,Y+	BF	(mmnn)	DD	(mmnn,PC)[3]	FC	(nn,PC)[3]
A1	,Y+	C0	,U+	DF	(mmnn)	FD	(mmnn,PC)[3]
A2	,-Y	C1	,U++	E0	,S+	FF	(mmnn)

注:

1. 操作数所表示的寻址方式的情况参考附录6。
2. 标号，PCR这种形式，可以在源程序清单中使用。
3. 〔标号，PCR〕这种形式，可以在源程序清单中使用。

附录10　6809、6829、6839、6842、6821、6850简明资料

附10.1　6809

附表10.1　6809机器码的十六制进数值

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	•				31	LEAY		4+	2+	61	•			
02	•				32	LEAS		4+	2+	62	•			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Inherent	5+	2	64	LSR		6+	2+
05	•				35	PULS		5+	2	65	•			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	•				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	•				3B	RTI		6, 15	1	6B	•			
0C	INC		6	2	3C	CWAI		20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL		11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	•				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Inherent	19	1	6F	CLR	Indexed	6+	2+
10	Page 2	–	–	–	40	NEGA	Inherent	2	1	70	NEG	Extended	7	3
11	Page 3	–	–	–	41	•				71	•			
12	NOP	Inherent	2	1	42	•				72	•			
13	SYNC	Inherent	2	1	43	COMA		2	1	73	COM		7	3
14	•				44	LSRA		2	1	74	LSR		7	3
15	•				45	•				75	•			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	•				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Inherent	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	•	–			4B	•				7B	•			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Inherent	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG		8	2	4E	•				7E	JMP		4	3
1F	TFR	Inherent	6	2	4F	CLRA	Inherent	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Inherent	2	1	80	SUBA	Indexed	2	2
21	BRN		3	2	51	•				81	CMPA		2	2
22	BHI		3	2	52	•				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	•				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRA		2	1	87	•			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	•				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	•				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Inherent	2	1	8F	•			

inherent: 固有
immed: 立即
Direct: 直接
Extended: 扩充
Indexed: 变址
Relative: 相对

符号:
- ~　MPU周期数（进出堆栈或变址方式周期可能少）
- #　程序字节数
- •　表示未用操作码

续表

OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA		4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA		4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX		5+	2+
AF	STX	Indexed	5+	2+
B0	SUBA	Extended	5	3
B1	CMPA		5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX		6	3
BF	STX	Extended	6	3
C0	SUBB	Immed	2	2
C1	CMPB		2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	Immed	2	2

OP	Mnem	Mode	~	#
C6	LDB	Immed	2	2
C7	•			
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	•			
CE	LDU	Immed	3	3
CF	•			
D0	SUBB	Direct	4	2
D1	CMPB		4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU		5	2
DF	STU	Direct	5	2
E0	SUBB	Indexed	4+	2+
E1	CMPB		4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	Indexed	5+	2+
F0	SUBB	Extended	5	3
F1	CMPB		5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	Extended	5	3

OP	Mnem	Mode	~	#
FC	LDD	Extended	6	3
FD	STD		6	3
FE	LDU		6	3
FF	STU	Extended	6	3

Page 2 and 3 Machine Codes

OP	Mnem	Mode	~	#
1021	LBRN	Relative	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS, LBCC		5(6)	4
1025	LBCS, LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	[illegible]
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT		5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI2	Inherent*	20	2
1083	CMPD	Immed	5	4
108C	CMPY		5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI3	Inherent	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

注：所有未用操作码都是未定义的和非法的.



附表10.2 6809指令系统简表

指令/形式		6809寻址方式																						
		固有			直接			扩充			立即			变址			相对				5	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~[5]	#	DE. 说明 ION	H	N	Z	V	C
ABX		3A	3	1																B + X → X 无符号	•	•	•	•	•
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A + M + C → A	↕	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A + M → A	↕	↕	↕	↕	↕
	ADDB				D3	4	2	FB	5	3	CB	2	2	EB	4+	2+				B + M → B	↕	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D + M:M + 1 → D	↕	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A ∧ M → A	•	↕	↕	0	•
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B ∧ M → B	•	↕	↕	0	•
	ANDCC										1C	3	2							CC ∧ IMM → CC					1
ASL	ASLA	48	2	1																A }	8	↕	↕	↕	↕
	ASLB	58	2	1																B } ← 0	8	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+				M } c b7 b0	8	↕	↕	↕	↕
ASR	ASRA	47	2	1																A }	8	↕	↕	•	↕
	ASR	57	2	1																B } →	8	↕	↕	•	↕
	ASR				07	6	2	77	7	3				67	6+	2+				M } b7 b0 C	8	↕	↕	•	↕
BCC	BCC																24	3	2	C = 0, 转	•	•	•	•	•
	LBCC																10 24	5(6)	4	C = 0, 长转	•	•	•	•	•
BCS	BCS																25	3	2	C = 1, 转	•	•	•	•	•
	LBCS																10 25	5(6)	4	C = 1, 长转	•	•	•	•	•
BEQ	BEQ																27	3	2	Z = 0, 转	•	•	•	•	•
	LBEQ																10 27	5(6)	4	Z = 0, 长转	•	•	•	•	•
BGE	BGE																2C	3	2	≥0, 转	•	•	•	•	•
	LBGE																10 2C	5(6)	4	≥0, 长转	•	•	•	•	•
BGT	BGT																2E	3	2	>0, 转	•	•	•	•	•
	LBGT																10 2E	5(6)	4	>0, 长转	•	•	•	•	•
BHI	BHI																22	3	2	大于，转	•	•	•	•	•
	LBHI																10 22	5(6)	4	大于，长转	•	•	•	•	•
BHS	BHS																24	3	2	大于等于，转	•	•	•	•	•
	LBHS																10 24	5(6)	4	大于等于，长转	•	•	•	•	•
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				位测A	•	↕	↕	0	•
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				位测B	•	↕	↕	0	•
BLE	BLE																2F	3	2	≤0, 转	•	•	•	•	•
	LBLE																10 2F	5(6)	4	≤0, 长转	•	•	•	•	•
BLO	BLO																25	3	2	小于，转	•	•	•	•	•
	LBLO																10 25	5(6)	4	小于，长转	•	•	•	•	•
BLS	BLS																23	3	2	小于等于，转	•	•	•	•	•
	LBLS																10 23	5(6)	4	小于等于，长转	•	•	•	•	•
BLT	BLT																2D	3	2	< 0, 转	•	•	•	•	•
	LBLT																10 2D	5(6)	4	< 0, 长转	•	•	•	•	•
BMI	BMI																2B	3	2	负，转	•	•	•	•	•
	LBMI																10 2B	5(6)	4	负，长转	•	•	•	•	•
BNE	BNE																26	3	2	Z≠0, 转	•	•	•	•	•
	LBNE																10 26	5(6)	4	Z≠0, 长转	•	•	•	•	•

续表

指令/形式		固有			直接			扩充			立即			变址			相对			说明	5	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	N	Z	V	C
BPL	BPL																2A	3	2	正，转	•	•	•	•	•
	LBPL																10	5(6)	4	正，长转	•	•	•	•	•
																	2A								
BRA	BRA																20	3	2	转	•	•	•	•	•
	LBRA																16	5	3	长转	•	•	•	•	•
BRN	BRN																21	3	2	不转	•	•	•	•	•
	LBRN																10	5	4	不长转	•	•	•	•	•
																	21								
BSR	BSR																8D	7	2	转子	•	•	•	•	•
	LBSR																17	9	3	长转子	•	•	•	•	•
BVC	BVC																28	3	2	V = 0，转	•	•	•	•	•
	LBVC																10	5(6)	4	V = 0，长转	•	•	•	•	•
																	28								
BVS	BVS																29	3	2	V = 1，转	•	•	•	•	•
	LBVS																10	5(6)	4	V = 1，不转	•	•	•	•	•
																	29								
CLR	CLRA	4F	2	1																0 → A	•	0	1	0	0
	CLRB	5F	2	1																0 → B	•	0	1	0	0
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0 → M	•	0	1	0	0
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				M同A比	8	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				M同B比	8	↕	↕	↕	↕
	CMPD				10	7	3	10	8	4	10	5	4	10	7+	3+				M:M + 1	•	↕	↕	↕	↕
					93			B3			83			A3						同D比					
	CMPS				11	7	3	11	8	4	11	5	4	11	7+	3+				M:M + 1	•	↕	↕	↕	↕
					9C			BC			8C			AC						同S比					
	CMPU				11	7	3	11	8	4	11	5	4	11	7+	3+				M:M + 1	•	↕	↕	↕	↕
					93			B3			83			A3						同U比					
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				M:M + 1	•	↕	↕	↕	↕
																				同X比					
	CMPY				10	7	3	10	8	4	10	5	4	10	7+	3+				M:M + 1	•	↕	↕	↕	↕
					9C			BC			8C			AC						同Y比					
CCM	COMA	43	2	1																$\bar{A}$ → A	•	↕	↕	0	1
	COMB	53	2	1																$\bar{B}$ → B	•	↕	↕	0	1
	OOM				03	6	2	73	7	3				63	6+	2+				$\bar{M}$ → M	•	↕	↕	0	1
CWAI		3C	20	2																CC ∧ IMM → CC 等待中断					1
DAA		19	2	1																十进制调整A	•	↕	↕	0	↕
DEC	DECA	4A	2	1																A − 1 → A	•	↕	↕	↕	•
	DECB	5A	2	1																B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1 → M	•	↕	↕	↕	•
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A ⊻ M → A	•	↕	↕	0	•
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B ⊻ M → B	•	↕	↕	0	•
EXG	R1 R2	1E	7	2																R1 ↔ R2′	•	•	•	•	•
INC	INCA	4C	2	1																A + 1 → A	•	↕	↕	↕	•
	INCB	5C	2	1																B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA³ → PC	•	•	•	•	•
JSR					9D	7	2	BD	8	3				AD	7+	2+				跳转子	•	•	•	•	•
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M → A	•	↕	↕	0	•
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M → B	•	↕	↕	0	•
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M:M + 1 → Q	•	↕	↕	0	•
	LOS				10	6	3	10	7	4	10	4	4	10	6+	3+				M:M + 1 → S	•	↕	↕	0	•
					DE			FE			CE			EE											
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M:M + 1 → U	•	↕	↕	0	•
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M:M + 1 → X	•	↕	↕	0	•
	LDY				10	6	3	10	7	4	10	4	4	10	F+	3+				M:M + 1 → Y	•	↕	↕	0	•
					9E			BE			8E			AE											
LEA	LEAS													32	4+	2+				EA³ → S	•	•	•	•	•
	LEAU													33	4+	2+				EA³ → U	•	•	•	•	•
	LEAX													30	4+	2+				EA³ → X	•	•	↕	•	•
	LEAY													31	4+	2+				EA³ → Y	•	•	↕	•	•
LSL	LSLA	48	2	1																A	•	↕	↕	↕	↕
	LSLB	58	2	1																B } C ← b7 … b0 ← 0	•	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+				M	•	↕	↕	↕	↕

续表

指令/形式		固有			直接			扩充			立即			变址			相对			说明	5	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~[5]	#		H	N	Z	V	C
LSR	LSRA	44	2	1																A} 0→b7…b0→C	•	0	↕	•	↕
	LSRB	54	2	1																B}	•	0	↕	•	↕
	LSR				04	6	2	74	7	3				64	6+	2+				M}	•	0	↕	•	↕
MUL		3D	11	1																A×B→D (无符号)	•	•	↕	•	9
NEG	NEGA	40	2	1																$\overline{A}$+1→A	8	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$+1→B	8	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$+1→M	8	↕	↕	↕	↕
NOP		12	2	1																空操作	•	•	•	•	•
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A∨M→A	•	↕	↕	0	•
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B∨M→B	•	↕	↕	0	•
	ORCC										1A	3	2							CC∨IMM→CC					7
PSH	PSHS	34	5+[4]	2																进S栈	•	•	•	•	•
	PSHU	36	5+[4]	2																进U栈	•	•	•	•	•
PUL	PULS	35	5-[4]	2																出S栈	•	•	•	•	•
	PULU	37	5-[4]	2																出U栈	•	•	•	•	•
ROL	ROLA	49	2	1																A} C b7 ← b0	•	↕	↕	↕	↕
	ROLB	59	2	1																B}	•	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M}	•	↕	↕	↕	↕
ROR	RORA	46	2	1																A} C b7 → b0	•	↕	↕	•	↕
	RORB	56	2	1																B}	•	↕	↕	•	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M}	•	↕	↕	•	↕
RTI		3B	6/15	1																中断返回					7
RTS		39	5	1																子程序返回	•	•	•	•	•
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A－M－C→A	8	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B－M－C→B	8	↕	↕	↕	↕
SEX		1D	2	1																扩B符号进A	•	↕	↕	0	•
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A→M	•	↕	↕	0	•
	STB				D7	4	2	F7	5	3				E7	4+	2+				B→M	•	↕	↕	0	•
	STD				DD	5	2	FD	6	3				ED	5+	2+				D→M; M+1	•	↕	↕	0	•
	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S→M; M+1	•	↕	↕	0	•
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M; M+1	•	↕	↕	0	•
	STX				9F	5	2	BF	6	3				AF	5+	2+				X→M; M+1	•	↕	↕	0	•
	STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y→M; M+1	•	↕	↕	0	•
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A－M→A	8	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B－M→B	8	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D－M:M+1→D	•	↕	↕	↕	↕
SWI	SWI[6]	3F	19	1																SWI_1	•	•	•	•	•
	SWI2[6]	10 3F	20	2																SWI_2	•	•	•	•	•
	SWI3[6]	11 3F	20	2																SWI_3	•	•	•	•	•
SYNC		13	≥2	1																同步中断	•	•	•	•	•
TFR	R1 R2	1F	7	2																R1→R2[2]	•	•	•	•	•
TST	TSTA	4D	2	1																测A	•	↕	↕	0	•
	TSTB	5D	2	1																测B	•	↕	↕	0	•
	TST				0D	6	2	7D	7	3				6D	6+	2+				测M	•	↕	↕	0	•

变址寻址 种类	形式	直接 汇编符号	后缀字节操作制	+~	+#	间接 汇编符号	后缀字节操作制	+~	+#
用R作常数偏值	无偏值	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5位偏值	n,R	0RRnnnnn	1	0	无8位			
	8位偏值	n,R	1RR01000	1	1	[n,R]	1RR11000	4	1
	16位偏值	n,R	1RR01001	4	2	[n,R]	1RR11001	7	2

变址寻址方式　　　　　　　　　　　　　　　　　　　　　　　　续表

用R作累加器偏值	A寄存器偏值	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B寄存器偏值	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	C寄存器偏值	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
R自动增/减	加1	R+	1RR00000	2	0	不允许			
	加2	R++	1RR00001	3	0	[, R++]	1RR10001	6	0
	减1	−R	1RR00010	2	0	不允许			
	减2	−−R	1RR00011	3	0	[, −−R]	1RR10011	6	0
用PC作常数偏值	8位偏值	n, PCR	1XX01100	1	1	[n, PCR]	1XX11100	4	1
	16位偏值	n PCR	1XX01101	5	2	[n, PCR]	1XX11101	8	2
间接扩充	16位地址	—	—	-	-	[n]	10011111	5	2

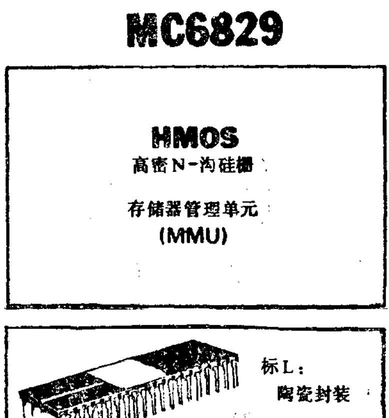
R＝X、Y、U、S
X＝任意

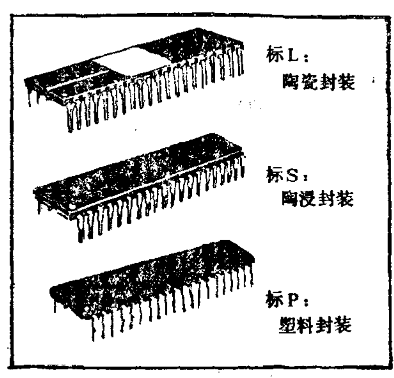
附10.2　6829 存储器管理单元（MMU）

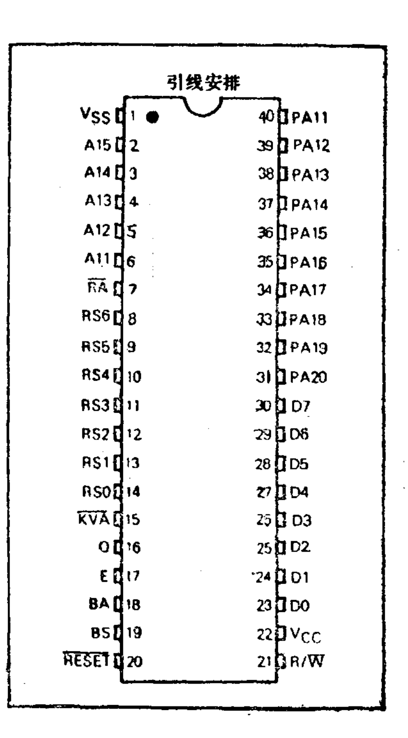
6829存储器管理单元的工作原理是使6809的地址空间从64K字节最大可扩充到2M字节。每个MMU可以同时处理包括DMA在内的四个不同的并发任务。MMU还可以保护一个任务的地址空间不会被另一个任务来修改。存储器地址空间的扩充方法是采用处理器的高五位地址线（A11～A15）和5位任务寄存器的内容共同来控制内部高速地址分配RAM而实现的。MMU输出端有10条地址线PA11～PA20，当其同处理器的低11条地址线A0～A10组合在一起时，即可以形成地址空间为2M字节。每个任务都可以给以2K字节为单位增量的存储器页面，一直可以用到全部64K字节。使用这种方法时，不同任务的地址空间对另一个任务来说完全可以保持独立。所以由于地址空间程序模式的简单化，大大提高了复杂的多处理系统软件可靠性。

其特点是：

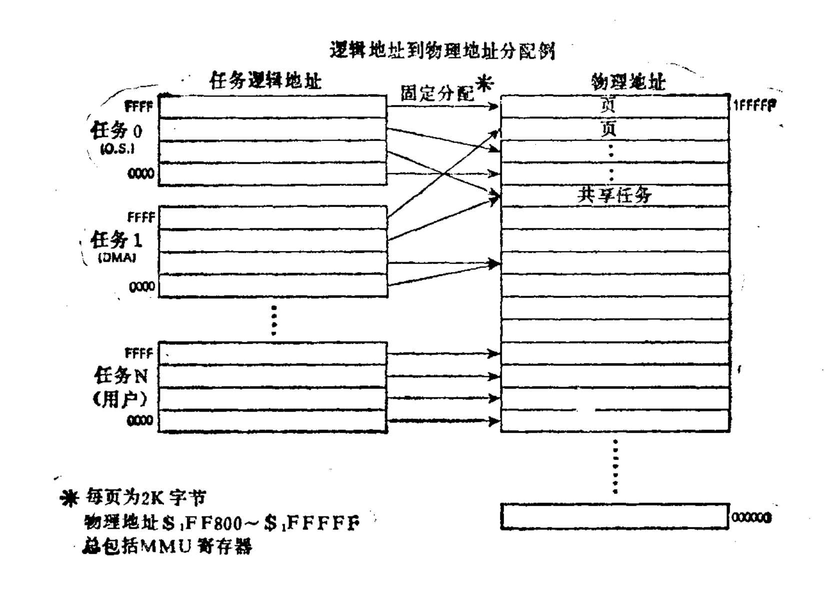
（1）存储器地址空间可从64K字节扩充到2M字节；
（2）每个MMU都具有处理单独四个任务的能力；
（3）系统中可以使用8个MMU器件；
（4）提供任务隔离和写保护；
（5）提供有效的存储器分配方式，每页2K字节，有1024页；
（6）设计时考虑到有效地使用DMA；
（7）器件内具有快速、自动切换任务能力；
（8）允许通过共享资源实现进程间相互通信；
（9）简化了程序设计的地址空间模式；
（10）提高了系统软件可靠性；
（11）同6809/6800系列总线兼容；
（12）单一＋5V电源。



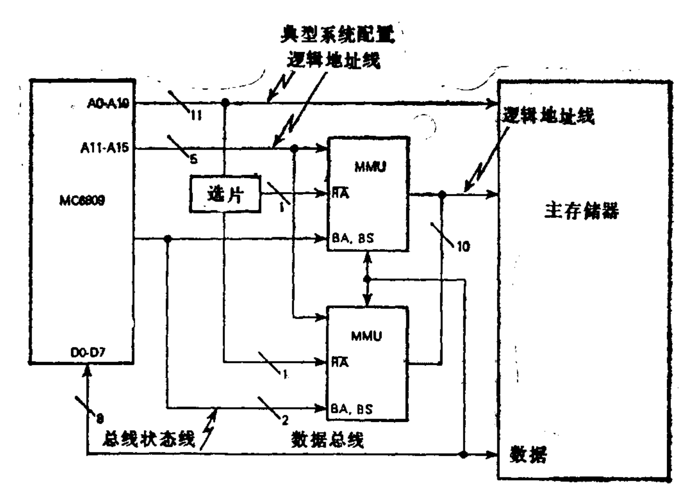




附图10.1　6829器件外形及引线



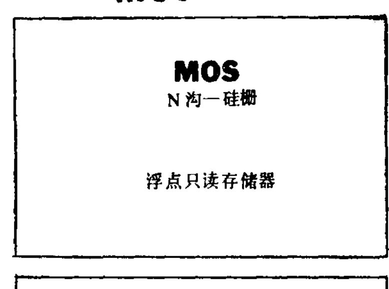
附图10.2　6829逻辑地址变换到物理地址的空间分配举例

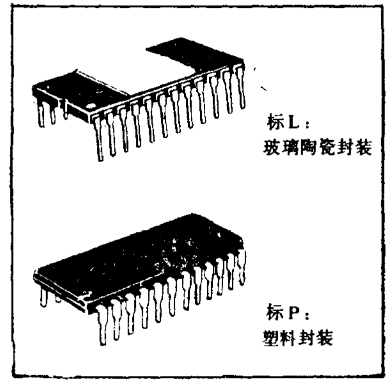


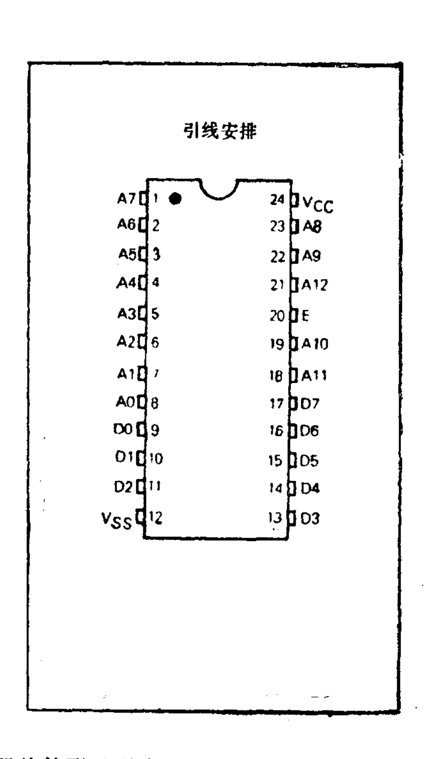
附图10.3　6829的典型系统配置

附10.3　6839——6809用的浮点ROM









附图10.4　6839器件外形及引线

特点：

（1）完全位置独立；

（2）不用绝对地址的RAM（重入）；

（3）操作数在寄存器中或堆栈中（Pascal）；

（4）同建议的IEEE标准兼容；

（5）单、双和双扩充格式；

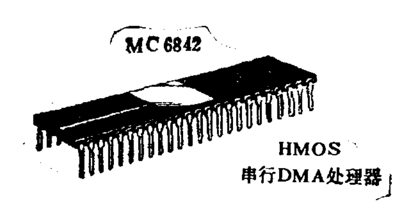
（6）有以下运算操作：

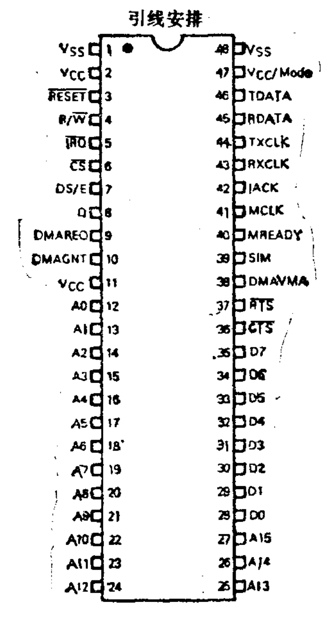
加法	求整数部分
减法	求绝对值
乘法	变反码
除法	比较
求余数	整数和浮点相互变换
求平方根	二进制和十进制相互变换

附10.4　6842串行DMA处理器

6842串行直接存储器存取处理器（SDMA）可以在微处理器和分布处理系统中的智能控制器之间，进行高速的串行数据交换。使用IBM公司制定的同步数据链路控制（SDLC）协议，使其具有多点、点-点或循环配置进行通信处理的能力。同时还支持许多种HDLC协议的要求。

SDMA 可接收局部微处理器的命令，并传送数据或发出链接电平的命令。SDMA 可以发出并响应多个链接电平命令，保证数据的完整性和有效性，以及对某些错误复原的处理。





附图10.5　6842器件外形及引线

其特点是：

（1）传输速率每秒可达4 M位；

（2）外部数据恢复；

（3）外时钟；

（4）DMA命令和数据链；

（5）SDLC协议；

（6）HDLC；

（7）全双工或半双工操作；

（8）发送和接收各有单独的DMA通道；

（9）正常的或系统地址检测；

（10）同6809、6800总线兼容；

（11）NRZ/NRZI数据；

（12）内部设位同步和字节同步；

（13）点-点、多点和循环方式；

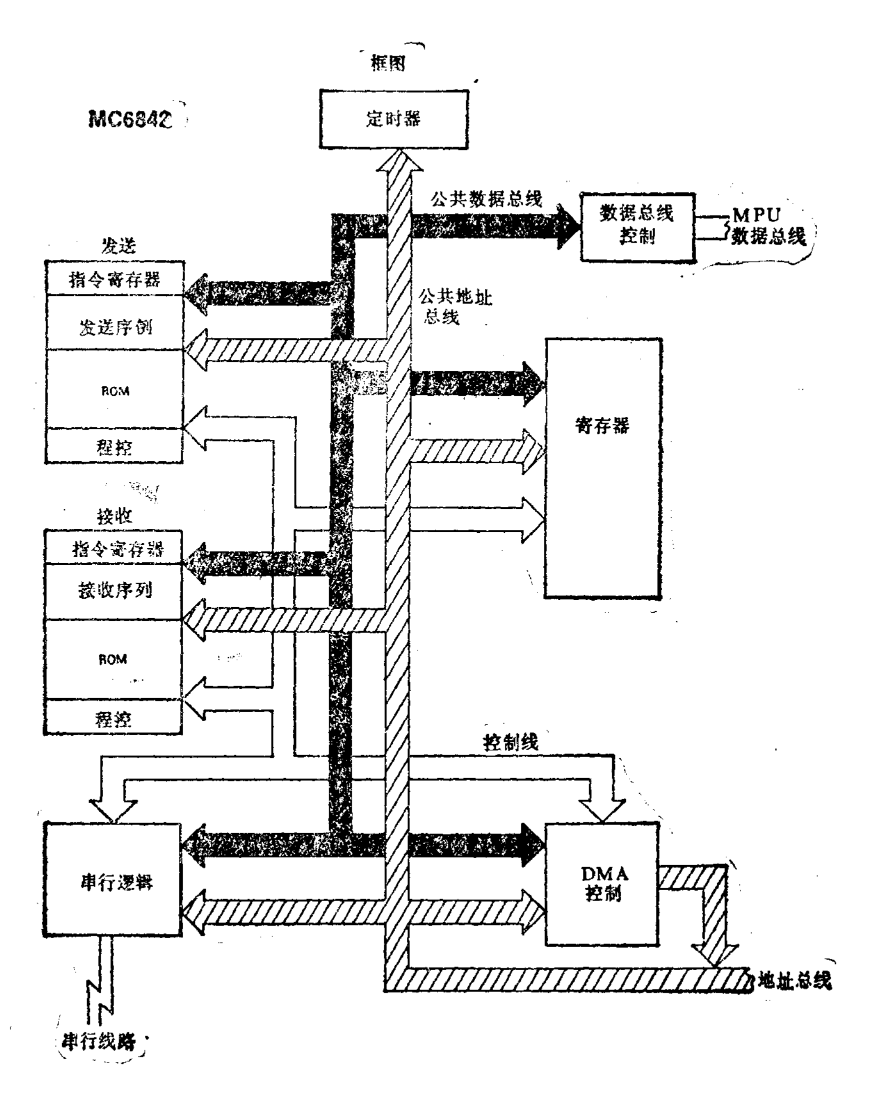
（14）CRC产生器和检验；

（16）主/次配置

（16）循环方式时，使用外电源；

（17）设初始化控制引线（SIM）；

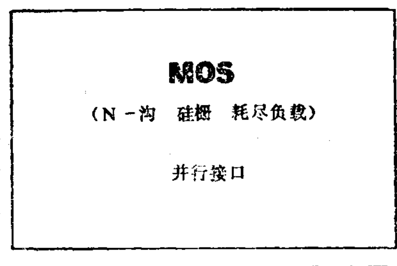
（18）有向量中断能力（IACK）。

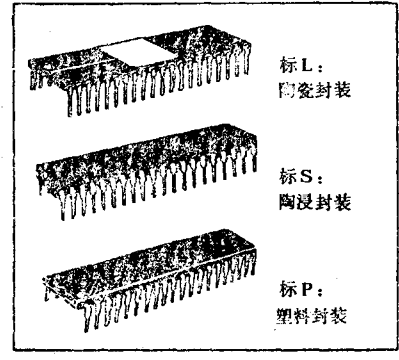


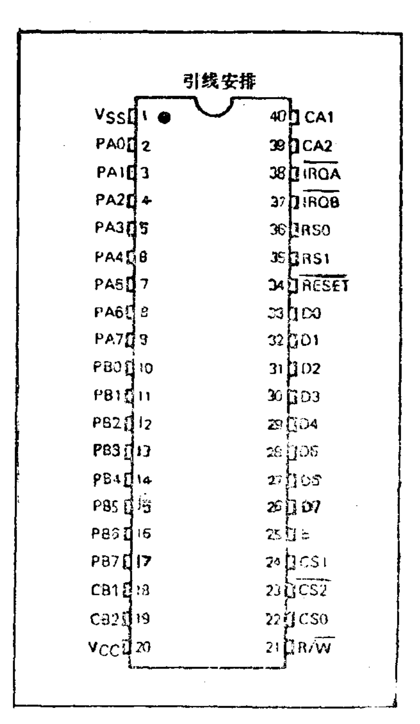
附图10.6　6842内部框图

附10.5　6821 并行接口控制器

MC6821







附图10.7　6821器件外形和引线

PIA 中有 6 个单元可以接收 MPU 数据总线的数据：2 个外设寄存器、2 个数据方向寄存器和 2 个控制寄存器。对这些单元的选择由 RS 0 和 RS1 输入端以及控制寄存器的第 2 位一起进行控制，如附表10.3所示。

附表10.3　内部寻址

RS1	RS0	控制寄存器第 2 位		被选单元
		CRA－2	CRB－2	
0	0	1	×	外设寄存器 A
0	0	0	×	数据方向寄存器 A
0	1	×	×	控制寄存器A
1	0	×	1	外设寄存器 B
1	0	×	0	数据方向寄存器 B
1	1	×	×	控制寄存器 B

×＝任意值

1. 初始化

$\overline{\text{RESET}}$线可以清零PIA所有的寄存器。这时，将使PA 0 ～PA 7 、PB 0 ～PB 7 、CA 2 和CB 2 置为输入端，而且所有中断被屏蔽。在$\overline{\text{RESET}}$总清信号之后执行再起动程序期间，PIA 必须要进行安排。

数据方向和控制寄存器的可能的详细安排如下内容所述。

2. 数据方向寄存器（DDRA和DDRB）

这两个数据方向寄存器允许MPU控制通过每一条外设数据线传输的数据方向。数据方向寄存器为0时，相当于外设数据线作为输入端；为1时，作为输出端。

3． 控制寄存器（CRA和CRB）

CRA和CRB这两个控制寄存器可使MPU控制四条外设控制线：CA 1、CA 2、CB 1和CB 2的操作。另外，这两个控制寄存器还可使MPU去启动中断线和监视中断标志位的状态。CRA和CRB中的第0位到第5位，当加上正确的片选信号和寄存器选择信号时，可以由MPU来读出或写入。其中第6位和第7位的内容只能读出，而且需由出现在CA 1、CA 2、CB 2或CB 2控制线上的外部中断信号进行修改。控制字的格式如附表10.4所示。

附表10.4 控制字格式

	7	6	5	4	3	2	1	0
CRA	IRQA 1	IRQA 2	CA 2控制			DDRA存取	CA 1控制	
CRB	IRQB 1	IRQB 2	CB 2控制			DDRA存取	CB 1控制	

（1）数据方向存取控制位（CRA-2和CRB-2）

CRA和CRB控制寄存器的第2位，在RS 0和RS 1端加上寄存器选择信号时，它就可以对外设接口寄存器或数据方向寄存器进行选择。

（2）中断标志位（CRA-6、CRA-7，CRB-6、CRB-7）

这四个中断标志位由四条中断线或外设控制线上信号的作用沿进行置位，这时，这些线应处在输入状态。这些中断标志不能直接由MPU数据总线来置位，但可以在适当时候，使用读外设数据的操作而被间接清零。

（3）CA 1和CB 1中断输入线的控制（CRA-0、CRB-0、CRA-1、CRB-1）

控制寄存器的最低两位可以控制中断输入线CA 1和CB 1。CRA-0和CRB-0位分别作为启动MPU中断信号$\overline{IRQA}$和$\overline{IRQB}$的控制；CRA-1和CRB-1位确定中断输入信号CA 1和CB 1的有效沿，见附表10.5。

附表10.5 中断输入CA1和CB1的控制

CRA-1 (CRB-1)	CRA-0 (CRB-0)	中断输入 CA 1(CB 1)	中断标志位 CRA-7(CRB-7)	MPU中断请求 $\overline{IRQA}(\overline{IRQB})$
0	0	↓有效	在CA1(CB1)↓处置为高电平	被禁止——$\overline{IRQ}$保持高电平
0	1	↓有效	在CA1(CB1)↓处置为高电平	当中断标志位CRA-7（CRB-7)到高电平时，变低电平
1	0	↑有效	在CA1(CB1)↑处置为高电平	被禁止——$\overline{IRQ}$保持高电平
1	1	↑有效	在CA1(CB1)↑处置为高电平	当中断标志位CRA-7(CRB-7)到高电平时，变低电平

注：

1. ↑表示正沿（从低电平到高电平）。
2. ↓表示负沿（从高电平到低电平）。
3. 中断标志位CRA-7用MPU读A数据寄存器来清零；而CRB-7用MPU读B数据寄存器来清零。
4. 如果CRA-0（CRB-0）为低电平，当中断出现（中断被禁止），而以后为高电平时，则在CRA-0（CRB-0）被写为"1"后，出现$\overline{IRQA}$（$\overline{IRQB}$）。

（4）CA 2 和 CB 2 外设控制线的控制（CRA-3、CRA-4、CRA-5、CRB-3、CRB-4、和CRB-5）

CRA和CRB这两个控制寄存器的第 3 、 4 、 5 位是对外设控制线CA 2 和 CB 2 进行控制的数字位。这些控制位确定控制线作中断输入线使用时的条件，以及作为输出控制信号使用时的条件。如果CRA- 5 （CRB- 5）位为低电平，则CA 2 （CB 2）将作为中断输入线使用，类似于CA 1 （CB 1），见附表10.6。当CRA- 5 (CRB- 5)为高电平时，则CA 2 (CB 2) 将作为输出信号端，可控制外设数据传送。当在输出方式时，CA 2 和 CB 2 在特性上稍有不同，见附表10.7和附表10.8。

附表10.6 CA2和CB2作为中断输入的控制

〔CRA-5（CRB-5）为低电平〕

CRA-5 (CRB-5)	CRA-4 (CRB-4)	CRA-3 (CRB-3)	中断输入 CA2(CB2)	中断标志位 CRA-6(CRB-6)	MPU中断请求 $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓有效	在CA2(CB2)↓处置为高电平	被禁止——IRQ保持为高电平
0	0	1	↓有效	在CA2(CB2)↓处置为高电平	当中断标志位CRA-6(CRB-6)到高电平时，变低电平
0	1	0	↑有效	在CA2(CB2)↑处置为高电平	被禁止——$\overline{IRQ}$保持为高电平
0	1	1	↑有效	在CA2(CB2)↑处置为高电平	当中断标志位CRA-6(CRB-6)到高电平时，变低电平

注：1. ↑表示正沿（从低电平到高电平）。

2. ↓表示负沿（从高电平到低电平）。

3. 中断标志位CRA- 6 用MPU读A数据寄存器来清零；而CRB- 6 用MPU读B数据寄存器来清零。

4. 如果CRA- 3 （CRB- 3）为低电平，出现了 CA 2 中断（中断被禁止），而以后为高电平时，即CRA- 3 (CRB- 3）被写为 1 后，则产生$\overline{IRQA}$（$\overline{IRQB}$）。

附表10.7 CB2作为输出的控制（CRB-5为高电平）

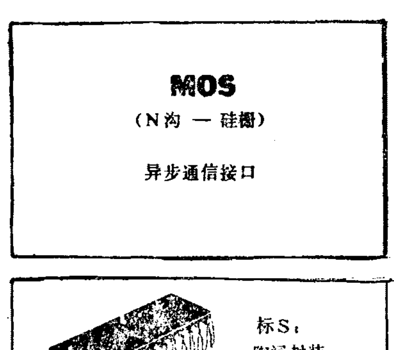
CRB-5	CRB-4	CRB-3	CB2	
			置 0	置 1
1	0	0	在MPU写“B”数据寄存器操作后的第一个E脉冲正沿，变为低电平。	当中断标志位CRB-7由CB1信号的有效沿置为1时，变为高电平。
1	0	1	在MPU写“B”数据寄存器操作后的第一个E脉冲正沿，变为低电平。	跟在第一个E脉冲后的E脉冲出现时的正沿处，变为高电平。
1	1	0	当MPU在控制寄存器“B”中使CRB-3写为低电平时，变为低电平。	只要CRB-3为低电平，总为低电平。在MPU写控制寄存器“B”使CRB-3为“1”时，变为高电平。
1	1	1	只要CRB-3为高电平，总为高电平。在MPU写控制寄存器“B”使CRB-3为“0”时，变为低电平。	当MPU写控制寄存器“B”使CRB-3为高电平时，变为高电平。

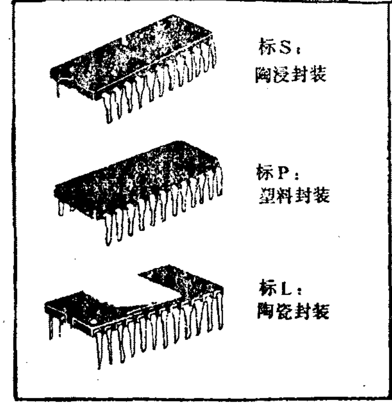
附表10.8　CA2作为输出的控制（CRA-5为高电平）

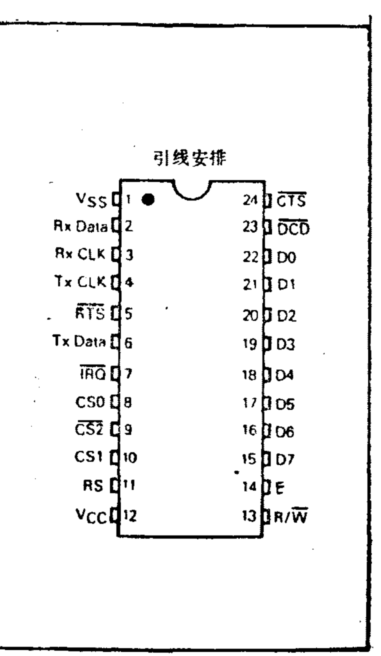
CRA-5	CRA-4	CRA-3	CA2	
			置　0	置　1
1	0	0	在MPU读"A"数据操作后E脉冲负沿，变为低电平。	当中断标志位CRA-7由CA1信号有效沿置为1时，变为高电平。
1	0	1	在MPU读"A"数据操作后E脉冲负沿，变为低电平。	在未选通期间出现的第一个E脉冲负沿，变为高电平。
1	1	0	当MPU写控制寄存器"A"使CRA-3为低电平时，变为低电平。	只要CRA-3为低电平，总为低电平。在MPU写控制寄存器"A"使CRA-3为"1"时，变为高电平。
1	1	1	只要CRA-3为高电平，总为高电平。在MPU写控制寄存器"A"使CRA-3为"0"时，变为低电平。	当MPU写控制寄存器"A"使CRA-3为高电平时，变为高电平。

附10.6　6850　异步串行通信接口控制器

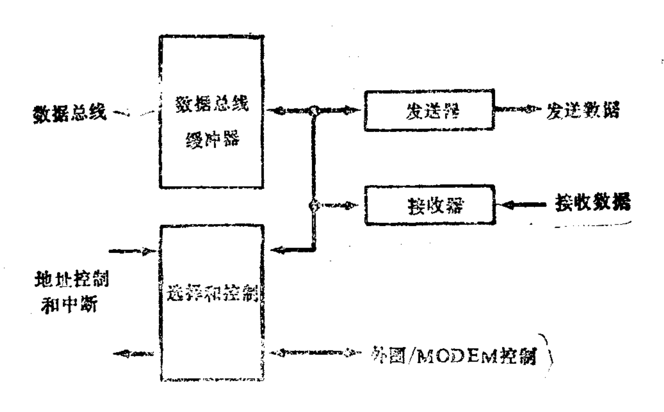
6850是一个通用异步通信接口控制器，它可以产生串行发送数据帧的格式，见附图10.8和附图10.9。该器件可以直接同6809总线相接，直接作为按存储器地址分配（映象）的I/O器件。数据流的转换可在两个方向进行，并可进行格式和错误的检测。由于传送的字长可变。具有中断控制、收发控制、调解器（MODEM）、控制以及时钟分频器的设定，因此，这是一个很通用的器件。该器件内部设有时钟发生器，但可以同MC1411时钟产生器一起使用。







附图10.8　6850器件外形和引线



附图10.9　6850异步通信接口简单框图

1．加电过程

6850器件内部设有总清电路，使在控制字中的总清位直到MPU使其置位为1之前保持为低电平。这样就保证在正确状态的瞬间之前不会工作。在确定其它控制位之前，必须使CR 0、CR 1这两位控制位确定好分频的数字。控制寄存器的选择，由MPU的控制线与适当的译码线结合来进行选择。如附表10.9所示。

2．控制寄存器

控制寄存器规定了ACIA的工作方式，在程序设计中必须要确定好。CR 0和CR 1控制时钟分频器和总清，见附表10.10。波特速率大多数情况下要看这两位置位的内容。CR 2、

附表10.9　ACIA中各寄存器内容规定

数据总线线号	缓冲寄存器地址			
	$RS\cdot\overline{R/W}$ 发送数据寄存器 (只写)	$RS\cdot R/W$ 接收数据寄存器 (只读)	$\overline{RS}\cdot\overline{R/W}$ 控制寄存器 (只写)	$\overline{RS}\cdot R/W$ 状态寄存器 (只读)
0	数据位0 *	数据位0	计数器分频选择1（CR 0）	接收数据寄存器满 (RDRF)
1	数据位1	数据位1	计数器分频选择2（CR 1）	发送数据寄存器空载 (TDRE)
2	数据位2	数据位2	字选择1（CR 2）	数据载频检测 (DCD)
3	数据位3	数据位3	字选择2（CR 3）	清除-发送 (CTS)
4	数据位4	数据位4	字选择3（CR 4）	数据帧格式错 (FE)
5	数据位5	数据位5	发送控制1（CR 5）	接收越界 (OVRN)
6	数据位6	数据位6	发送控制2（CR 6）	奇偶错 (PE)
7	数据位7 ***	数据位7 **	接收中断工作(CR 7)	中断请求 (IRQ)

*　发送时起始位＝0位
**　7位＋奇偶位方式中，该数据位为0
***　7位＋奇偶位方式中，该数据位为多余位，无用

附表10.10 分频选择位（CR0和CR1）

CR 1	CR 0	功　　能
0	0	÷1
0	1	÷16
1	0	÷64
1	1	总清

附表10.11 字选择位（CR2～CR4）功能

CR4	CR3	CR2	功　　能
0	0	0	7位＋偶数奇偶位＋2终止位
0	0	1	7位＋奇数奇偶位＋2终止位
0	1	0	7位＋偶数奇偶位＋1终止位
0	1	1	7位＋奇数奇偶位＋1终止位
1	0	0	8位＋2终止位
1	0	1	8位＋1终止位
1	1	0	8位＋偶数奇偶位＋1终止位
1	1	1	8位＋奇数奇偶位＋1终止位

附表10.12 发送状态（CR5、CR6）功能

CR6	CR5	
0	0	$\overline{RTS}$＝低电平，发送不能中断
0	1	$\overline{RTS}$＝低电平，发送可中断
1	0	$\overline{RTS}$＝高电平，发送不能中断
1	1	$\overline{RTS}$＝低电平，在发送数据输出端发送间隔，发送不能中断

CR 3和CR 4控制收发的数据形式。奇偶位种类、终止位和数据串的规定都在附表10.11中给出。CR 5和CR 6控制发送中断和请求发送线RTS，见附表10.12。如果使中断进行工作，发送数据寄存器空位时，则发送数据寄存器空位（TDRE）中断被置1。如果接收数据寄存器已满载，或者有一个数据串越界，或者在数据载波信号线上有一个上升沿存在的情况下，则CR 7为1，启动中断工作。

3. 状态寄存器

ACIA的状态寄存器可提供程序进行分支转移时使用的信息。这些数据位如果在工作状态下，它们可以启动规定的中断，或者它们可给执行的程序提供操作信息。第0位表示接收数据寄存器满，为当前数据占有，在读出或总清之后该位为0。DCD为高电平时，使第0位为低电平。第1位表示发送数据寄存器中的数据已被发送完毕，当前准备好发送下面的数据。第2位表示由MODEM来的载波检测存在，如果该位为1，则可产生中断请求。第2位可在读状态、或数据序列时被清零。在中断已被清除后，如果该位仍为高电平，则该位将随输入而变。第三位是清除-发送位，如果该位为0，则TDRE位停止工作，即有效地禁止发送。第4位表示数据帧格式错，任何时间都可用，接收缓冲区的字符串同该位有关。该位可以表示出同步出错、不合格的发送，或数据被截断搞乱。第5位是接收越界位，数据越界的条件是：在接收数据寄存器装满进入的字符即将完成之前——也就是在依次接收的第二个字

符的最后一位的终点时间，数据串中的前面的字符还没有被读出。这种越界错误在读出接收数据寄存器内容时即被清除。第6位是奇偶错指示位，它表示字符中“1”的个数不等于预定的奇数或偶数的和。只要数据处在接收数据寄存器之中，该位就表示有效。第7位是合成表示的中断位，任何一个中断都会反映到该位之中，在读出接收数据寄存器或写入发送数据寄存器时，该位都会被清除。

附录11 MEK6809EAC——MEK6809D4B 单板微型计算机用编辑汇编程序

MEK6809EAC是在MEK6809D4B单板微型计算机中使用的编辑汇编软件包。它可以记录到音频盒带上（以300波特），同MEK6809D 4的磁带格式兼容。

1. 一般性能

（1）可以固化在ROM之中；

（2）可以把6800和6809的记忆符汇编为6809的目标码：

（3）可以使用D4型单板机工作，也可以使用其它6809系统工作；

（4）可以自己定位存储器的容量；

（5）可以读出使用MEK68I/O的MEK6802EA 系统所产生的磁带；

（6）支持串行或并行打印机；

（7）交互会话式编辑程序；

（8）程序的存储可以使用MEK6809D 4的盒带；

（9）采用电视显示时，可以使用MEK68R 2 D。

2. 编辑命令及其性能

（1）可从音频盒带上增添文本文件；

（2）可把文本文件记录到音频盒带上；

（3）可在打印机上打印出文本文件；

（4）可查找文本文件的起点；

（5）对整个文本文件一次可以移动一个以上的字符；

（6）查找文本文件的终点；

（7）对整个文本文件一次可以移动一行以上的文本；

（8）插入/删除字符串；

（9）检索字符串；

（10）改变字符串；

（11）插入/删除一行；

（12）链接编辑命令。

3. 汇编程序命令及其性能

（1）汇编表格清单的速度控制允许在CRT屏幕上进行检查；

（2）程序清单表有分页和空间的控制；

（3）汇编程序可以控制伪操作码；

（4）可以在屏幕或打印机上产生汇编程序清单；

（5）有表格清单符号标志；

（6）目标码可以记录到音频盒带中；

（7）目标码可以输出到存储器中。

4. 存储器地址分配图

下面给出的存储器地址分配图是MEK6809 EAC的典型系统应用的情况。因为该程序可记录在音频盒带上，两面的带上有两种不同的地址安排，使用起来效果相同。具体地址安排如下：

1面

区域	高地址	低地址
D 4 BUG	FFFF	E800
I/O	E 0 FF	F000
CRT屏幕，更新RAM	9FFF	9000
可选文本缓冲区	8FFF	4000
标准文本缓冲区	3FFF	3000
编辑/汇编程序	2FFF	0100
暂存工作区	00FF	0000

2面

区域	高地址	低地址
D4BUG	FFFF	E800
I/O	E0FF	E000
编辑/汇编程序	CFFF	A000
CRT屏幕更新RAM	9FFF	8000
可选文本缓冲区	7FFF	2000
标准文本缓冲区	1FFF	033C
暂存工作区	033B	0000

5. 典型配置

MEK6809EA可以使用在各种系统配置中，但每种系统配置必须包括一些基本项目，它们是：6809型号的微处理器、监控程序ROM（D 4 BUG）、I/O设备以及足够数量的读/写存储器（RAM），才能完成编辑汇编程序的任务。

MEK6809D4B装备有6809MPU和所需的监控ROM。如果使用与RS-232C兼容的终端，MEK6809D4B也提供所需的I/O接口和程序。所用的终端也可以换成使用MEK68R 2 D接口板、CRT监视器（或改装的电视机）和ASCII编码键盘。

RAM存储器可以使用MEK68MM16（16K字节）或MEK68MM32（32K字节）的动态存储器模板。程序量较大时，可使用后一种的存储器板。

MEK6809EAC软件盒带的两面内容是一种编辑汇编版本，它适合于ROM或EPROM进行操作，因此给用户提供一种把编辑汇编程序写到EPROM中的机会，这时需把写了编辑汇编程序的EPROM插在MEK6809D 4 B上设置的用户ROM插座上使用（所需的EPROM可以使用3片4 K×8或者6片2 K×8的单电源EPROM。如果对MEK6809D 4 B板上的布线稍加修改也可以使用多电源的EPROM器件）。MEK6809D 4 B上设置的普通的用户RAM区插座是供小型程序设计使用的文本缓冲区空间，而且可以采用RS-232C终端组成单板微型计算机程序开发设备。

在多块单板系统配置中需要使用底板结构，这时可以使用MEK68CMB插件箱和母板组合件，也可以使用专门设有插件导轨的MEK68MB 5母板。如果选用廉价的MEK68MB 5，以后也可以升级到MEK68CC机箱。

附录12 MEK6809D4、MEK68KPD
单板机技术简介

1. 系统性能

（1）MEK6809D 4

- 使用MC6809高性能微处理器
- D 4 BUG监控程序固件（4 K）可扩充至 6 K
- 直接存储器存取
- 设备的中断
- 音频盒带接口，300波特或1200波特
- 可选用RS-232接口，设有6850ACIA
- 系统RAM512字节～1 K字节
- 用户RAM512字节～4 K字节
- RAM/ROM页面选择寄存器
- ROM地址分配技术
- 所有 I/O 和存储器全部被译码
- 停止地址比较器
- 设有内外部系统时钟
- 双向地址总线设有测试信号和控制逻辑
- 设有控制和状态线
- 系统缓冲器

（2）MEK68KPD

- 8 个 7 段显示器
- 25电键的按键板
- 板内设有电源
- 设有用户PIA MC6821
- 有绕接器件区
- 有16引线的辅助插座

2. 性能要点

（1）在 MEK 6809 D 4 单板机各部分之间以及单板和边上的插头接线之间都设有系统缓冲驱动器；

（2）硬件RAM和ROM 设置有选页面寄存器；

（3）设置的 4 K字节的静态用户RAM（8 个插座）可以 4 K字节为一块在 64K 字节的存储器空间，使用跳接器将其地址安排在任何地点。另外，还可以由 3 位硬件 RAM 页面寄存器进行控制跳接到所选的RAM页面；

（4）设有 8 个24引线的ROM插座，使用的 ROM/EPROM 的种类有 1 K × 8 单电源或三电源EPROM/ROM、2 K × 8 的单电源或三电源EPROM/ROM、4 K × 8 的EPROM

/ROM或者8 K×8的EPROM/ROM；

（5）采用按照ROM的地址分配技术，可在64K基本存储器空间以1 K为单位使8个ROM插座按照总的地址分配方式安排在任何地方。另外，各个插座的地址可由3位硬件ROM页面寄存器进行控制安排在任何ROM页面上；

（6）板上的所有存储器和I/O口都进行译码，所以在D 4板上没有特别需要的地址空间可以安排给另外单板上的存储器使用；

（7）在D 4板上因为可使用三电源EPROM，所以设有－12V～－5 V的调压器。必须由用户供给＋12V、－12V和＋5 V电压；

（8）设有盒带机接口硬件，使监控制程序中的软件可以存储和读取按照堪萨斯城标准的300波特或1200波特格式的盒带数据；

（9）设有停止地址比较器驱动的中断；

（10）由板上的3.579MHz的晶体作为系统时钟，或者是使用4倍频率的与TTL兼容的外部信号源提供系统时钟；

（11）外部的处理器可以通过70条引线的接头提供测试信号和逻辑，对板内的存储器和I/O器件进行控制和检查。

（12）设有控制线和状态线，增强了MPU和总线译码/驱动逻辑对硬件控制的灵活性。因此，可以实现以下功能：

- 测试和系统调试（硬件、软件）
- 完成各种中断（RESET、NMI、IRQ、FIRQ）
- 外部设备的中断向量（IVE、STKOP）
- 禁止中断（IRQE、FIRQE）
- 暂停和总线请求（BREQ）
- 增设慢速存储器（MEMRDY）
- 实现DMA

在MEK6809D 4 B中设有以下标准技术性能，而在MEK6809D 4 A中也可以选用这些标准性能：

- 设有与RS-232兼容的串行口使用的缓冲交接信号；
- 波特速率发生器可以提供110、300、600、1200、4800和9600波特的时钟信号；
- 在总线接口处的地址线、数据线和控制线均设有缓冲驱动器电路。

3. 单板类型

（1）MEK6809D 4

在MEK6809D 4 A型的单板机上面没有插入RS-232电路器件，也没有插入地址和数据缓冲驱动器件（指70条总线使用的地址和数据驱动器）。设计D 4 A型单板的目的是同MEK 68KPD按键板/显示器单元一起使用，而且该板设有电源，可使整个系统工作。同样在D4A板中也不提供“用户RAM”器件，只设有插座。为该单板机提供4 K系统监制程序。

在MEK6809D 4 B型单板机的主要目的是想使用RS-232C串行终端，或者使用MEK68 R 2 D单板作为系统终端的CRT接口。在D 4 B上设有RS-232C串行接口电路以及数据和地址总线驱动器。

为使RS-232C接口电路工作，必须加＋12V、＋5 V、－12V三种电源。设置4 K＋2 K

的监控制序。提供给用户的D4B中，在“用户RAM”区中也不供给RAM，需用户自己准备。

莫托罗拉公司生产的扩充单板系列M-60、M-70同D4B兼容，可以使用ASCII键盘接口到本微型计算机系统。MEK68KPD（带有电源）可以给MEK6809D4B作为键盘/显示器使用。

4．功能扩充件

目前，微处理器工业在飞速前进，因而客观上有一个迫切的要求就是提供学习和评价资料，以便帮助广大的工程技术人员迅速适应这种技术的发展。

为了满足这种要求，莫托罗拉Memory系统公司发展了一种单板系列，准备作为对M6800集成电路系列的学习和评价性产品。这套系列称为“MOKEP”即MOTOROL Kit Expansion Products（莫托罗拉单板系列扩充产品），其中有以下各种产品：

（1）MEK68CC插件箱

MEK68CC插件箱使用MEK68MB5母板。

（2）MEK68MB5母板组件

MEK68MB5母板备有中心距5/8英寸的10个插件槽，每个槽装有一个70条引线的插件座。

（3）MEK68CMB插件箱/母板

MEK68CMB可以装入10块MOKEP插件板，插件箱与MEK68CC相同。母板使用的是MEK68MB5那种形式，但没有单独的插件导轨。整个安装尺寸是：高为$8^1/_4$英寸，宽为$7^1/_4$英寸，深为$13^1/_4$英寸。

（4）MEK68R2/R2D/R2M程控CRT接口组件

MEK68R2/R2D/R2M程控CRT接口组件与MOKEP系列中其它产品一起可以构成微型计算机系统。MEK68R2D同MEK6809D4单板微型计算机及MEK68MB系列母板一起使用。上述所有组件特点是采用软件方法都可以程控行和字符的格式，按5×7点阵显示、半图形显示以至到4K存储器屏幕显示。所有组件都设有ASCII键盘接口。

（5）MEK68IO输入输出组件

MEK68IO是300/1200波特盒带接口，其中设有2个6850ACIA、一个MC14411波特率发生器和一个6821PIA。

（6）MEK68EP EPROM编程组件

MEK68EP是对单电源和三电源1K、2K、4K EPROM进行写入程序的组件。

（7）MEK68RR ROM/RAM组件

MEK68RR备有8个ROM插座，可以使用1K、2K、4K、8K单电源或三电源ROM或EPROM。同时该板上还装有8K字节的静态RAM插座。

（8）MEK68MM16/MM32 16K/32K存储器组件

MEK68MM16和MEK68MM32 分别为16K字节或32K字节的RAM。MEK68MM存储板使用16K动态RAM，利用隐更新技术保持存储单元内容，因此是成本低廉、功耗减小、密度较高的动态存储器系统，因为是采用内部隐含更新方法，从系统角度看和静态存储器使用方式一样。MEK68MM完全支持D4型微型计算机系统的RAM页面划分技术，在一个系统中允许使用8块存储板或者是256K字节RAM。

（9）MEK68WW/WW 1 绕接组件

设计MEK68WW绕接组件的目的是使MOKEP系列单板可以同其它性能的单板连接，以增强微型计算机系统的能力。MEK68WW可以同MEK6800AB连接器或母线板一起使用，并直接与AB的60条总线接口。MEK68WW 1 使用70条总线，直接同MEK68MB系列母板接口。同时，每种绕接组件还都可作为插件转接板。这两种组件还设有地址总线、数据总线和控制总线所需要的驱动器电路。

（10）MEK6809EA编辑/汇编程序

MEK6809EA编辑/汇编程序为MEK6809D 4 B用户提供在M6809上执行的输入、汇编、编辑和保存汇编语言程序的能力。编辑程序可以用来输入和编辑文本文件，准备进行汇编后执行的软件程序。该汇编程序可以接受M6800和M6809 两种记忆符号。由汇编程序产生的目标码可放在存储器之中或保存在磁带之上。该编辑/汇编程序也支持 MEK68R 2 D 显示接口板和单独的终端。

5. 软件性能

MEK6809D 4 操作系统为开发和操作用户的程序提供了方便的条件。基本监控程序与MEK68KPD按键板/显示器单元相接口，并放在 4 K字节的ROM之中（为MCM68332或相同类型器件）。出厂时，该ROM即装在MEK6809D 4 产品中，不再另外提供。

在MEK6809D 4 中还使用另外 2 K字节ROM（MCM-68316E或相同类型器件），作为同MEK68R 2 D CRT监视器和ASCII键盘接口，或者同兼容RS-232C的终端接口。这种附加的 2 K ROM只在MEK6809D 4 B产品之中提供给用户使用。

该监控程序的源程序清单和完整的内容资料可由莫托罗拉公司有偿提供给用户。监控程序本身是使用功能很强的子程序，并按位置独立代码程序编写出来的。这些源程序对许多用户程序的编制都是很有价值的。

监控程序有以下功能：

（1）检查/修改存储器单元内容

用户可以指定任何存储器单元并显示其内容，需要时对当前所选的单元可以输入新的数据。如果把数据写到了一个无效的单元时，那么，新的数据将同原来在该无效单元中的数据一起被显示出来。只有在完成正确的修改时，存储单元中新的数据才被显示出来。

在检查/修改存储器时，用户还可以自动地选择下面的或者是以前单元内容，完成之后可以退回到监控程序。

（2）检查/修改寄存器

该功能可以使用户检查/修改两个外部寄存器和6809中 9 个内部寄存器所对应的堆栈RAM存储单元区。因此这就等于允许用户来检查/修改这些寄存器单元中的内容。

该功能与存储器的检查/修改的区别是寄存器要按一套顺序来显示。而且目的寄存器和其内容都将被显示出来，这种功能用起来较为简单。

在MEK6809D 4 中一起显示的两个外部寄存器的操作内容不是6809所固有的。

（3）停止地址

在调试程序的过程中，当与规定的地址相符时，能够使机器暂停执行程序，这是很有利的。应用这种功能的一个典型的例子是：在用户程序运行过程中，需要确定某个存储器单元发生偶然变化的原因（不正确的改变）时，这种方法就会很有用。

MEK6809D 4 中实现停止地址功能的方法是采用使MPU地址输出同停止地址寄存器中用户所输入的数据相比较的线路完成。当比较结果相同时，就立即产生非屏蔽中断，从而实现停止在某地址的功能。

根据指令的类型，更准确地说，要根据指令周期中寻址所确定的时间关系，NMI可在前面指令的结尾被识别出来。然后控制转移到监控程序，使用户去确定二条特定指令之一已访问过规定的存储器单元。

在某些情况下，希望程序只停止在符合地址的第N个时间拍节，MEK6809D 4 可以实现这种功能。同时在每次地址符合时间还可以输出一个触发脉冲，而不是停止程序的执行。

（4）断点

实现停止在某地址上的功能采用的是硬件方法，而程序执行过程中被截断停止的软件方法就是要求在程序之中设置断点。其有效的方法就是用软件中断来代替在那个单元中的指令。

在用户程序中可以设置 8 个点（但程序需在RAM之中）。如使用停止地址方法时，由用户所能使用的断点就会失去N－1个（因为无论停止地址还是设置断点方式其最大N值为255）。

用户可以通过断点编辑程序的功能实现对断点的设置、删除或检查。

（5）跟踪指令

该指令允许用户执行程序时，每次前进一条指令。在每条指令结束时，自动地进入检查/修改寄存器的程序，并显示新的程序计数器内容。

（6）跟踪用户程序（线性跟踪）

如果把子程序作为一条指令来看时，经常希望跟踪程序时要一次通过一个程序。这种情况下很明显的一个例子就是要求所有的子程序在此之前应该完全彻底调试通过。MEK6809 D 4 在调试程序时可以实现跟踪用户程序，而一次通过子程序的方法有两种：

第一种方法也是软件方法，即在子程序中的每条指令都要进行比较，直到遇有紧跟在子程序后的指令为止。这样从子程序调用到返回的这部分程序就其跟踪功能来说，就可以看成一条指令。对于嵌套的子程序也可以自动地由监控程序来进行处理。

第二种跟踪用户程序（线性跟踪）的方法可采用停止地址的方法实现。这种方法比软件方法有一个优点就是子程序的执行是处在实时条件下进行的。这一点在调试依赖于时间的I/O程序时特别有用（尤其对很长的子程序更有意义，因为软件方法要极大地增加子程序运行时间）。该方法的缺点是程序执行时往往在返回后要继续做一条指令。

（7）用户程序控制的执行

在MEK6809D 4 的监控程序中还设有使用户转移（GO TO）、继续（Continue）或退出（Abort）用户程序的功能。

（8）计算偏值

在一般情况下或者在作程序修改时，经常需要计算从跳越转移或分支转移指令所在单元到目的单元的偏值。而且某些变址寻址方式的指令也要使用到相对偏值。在检查/修改存储器单元的操作中带有一种附加的功能很容易实现这种偏值计算要求。

计算过程是：用户设置偏值的单元、偏值命令的种类，然后输入所要求的目的地址。MEK6809D 4 计算所需偏值、并显示偏值内容，然后把该数据输入到适当的存储单元之中。

偏值计算方法可以实现对短偏值和长偏值的计算。

（9）音频盒带的记录/装入/校验

音频盒带接口采用改进型堪萨斯城标准方式，可以实现300或1200波特的传输工作。该接口可以完成记录、装入或对存储器校验的任何一种功能，操作时有的要选偏值，有的不要给偏值。这种功能对于按位置独立代码来写的用户程序是特别有用的。

6. MEK6809D 4 B型单板机中增加的软件功能

（1）存储器转储

使用存储器转储命令可以相当ASCII代码的方式给用户显示存储器数据块。显示格式取决于显示设备的配置情况，其间稍有不同。在给显示地址时，结束地址必须是一个大于起始地址的十六进制数，否则就发出警告，接着又给出要求设置新的起始地址命令。直到满足被规定好的地址范围时，转储命令才能执行下去。

（2）填充存储器内容

监控命令中设有用四个字节数据形式的方法来填入一块存储器区。象转储存储器内容的命令一样，使用时要给出起始和结束地址。

（3）检索存储器内容

该功能可以允许检索一个指定的存储器数据块，这时要使用一个4字节组成的标题，放到一个对应的4字节的屏蔽区。因为屏蔽区所有各位都是0，所以组成标题中的对应位是什么内容都可以。

在规定了屏蔽字节区之后，检索的功能就会在规定好的地址范围中进行。

每当按算法完成比较后，相符的第一个单元的地址就被显示出来。如果比较成功的地址清单被显示得太快，那时可以使用"ESCAPE"键使列清单表的显示暂停。

（4）移动存储器内容

该功能可以把在存储器中某个区域内的数据块移到另一个新的区域之中，又称存储器"搬家"。

和存储器转储方式一样，需要输入起始地址和结束地址，在末地址输入之后，则会出现要求输入新的起始地址，而该地址就是数据块要移动到那里的起始地址。

（5）输入ASCII字符

该功能可以使用户迅速地把ASCII的数据存到存储器之中，而不要去看决定每个ASCII字符所对应的十六进制数字。

在编写信息过程中，可以结合其它性能而方便使用，这时可以调用D 4 BUG中的子程序"PDATA"。

7. MEK6809D 4 技术说明

（1）CPU

该单板机CPU由MC6809高性能微处理器构成，设有3.579MHz的石英晶体，而且在MC6809同D 4 板上的其它线路接口都设有缓冲驱动器。

（2）ROM

ROM存储器系统由8个插座构成，它可以接受各种类型的ROM器件，可以使用单或三电源的1 K、2 K、4 K、8 K字节的ROM或EPROM。

配置ROM时，使用小型跳接开关，非常方便使用，无需任何工具。

在存储器空间中，对8个ROM的地址分配使用地址分配ROM即可实现（类似可编程逻辑阵列——PLA）。在D 4系统中，页面技术可以使用的ROM可达192K字节。

（3）RAM

有两组1K×4的RAM，一组是堆栈RAM；另一组是用户RAM。

堆栈RAM主要是给D 4的操作系统堆栈和工作暂存单元使用。另外还要有512字节作为用户RAM。该RAM在D 4系统中的存储器单元地址为$E400～$E7FF。

用户使用的RAM设有4K×8字节存储区，在D 4存储器的地址分配中可以利用跳接器定位在任何地址之上。

如果不用这8个用户RAM插座上的存储器也可以采用MOKEP系列中的MEK68MM存储器板，这样系统存储器可以扩大。

当拿掉另外的跳接器时，可以防止写入用户RAM（写保护），但堆栈RAM不受此影响。在需设有跳接器的地方，要设置对用户RAM的读写后才会正常进行。

（4）地址总线系统

在某些微处理器系统中，地址流是从微处理器经过缓冲驱动器再到母板总线之上。在D4系统中使用较为复杂的组织结构，可以禁止微处理器工作。这样就会使外部来到D4的地址可以访问单板中的部件并允许某些应用中使用DMA方式。

（5）数据总线系统

在D 4系统中使用有四个双向数据总线驱动器；对ROM驱动器、RAM/IO驱动器、板内外转接头驱动器、MPU驱动器等各个驱动器都设有启动和方向的输入端。控制逻辑要使这些驱动器在D 4板中各部分之间传送数据。

（6）停止地址电路

使用停止地址目的的是在到达某个地址时，可以启动要执行的用户程序。

要把准备停止的地址存起来，而且当板上地址总线各位相符合时，就会输出结果，从而使非屏蔽中断出现，使微处理器转到服务程序。

当地址符合出现时，不必产生NMI，在这种情况下，在测试点可用比较器输出来给振荡器提供一个触发信号。

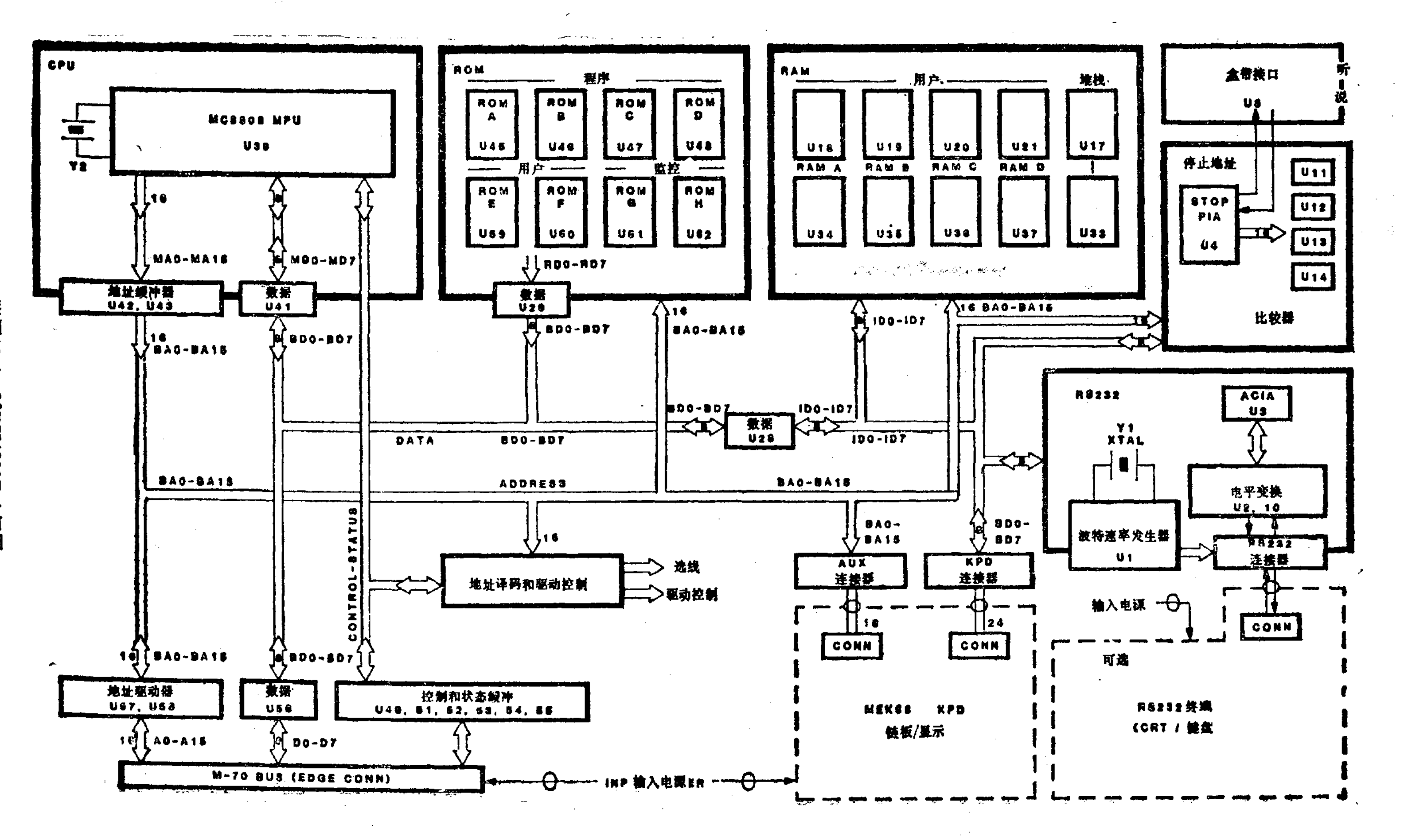
（7）RS-232电路

RS-232的技术指标规定了计算机与不同终端之间相互连接的标准。ACIA可以把总线上的并行数据转换为串行数据，然后把串行数据变换为RS-232电平。

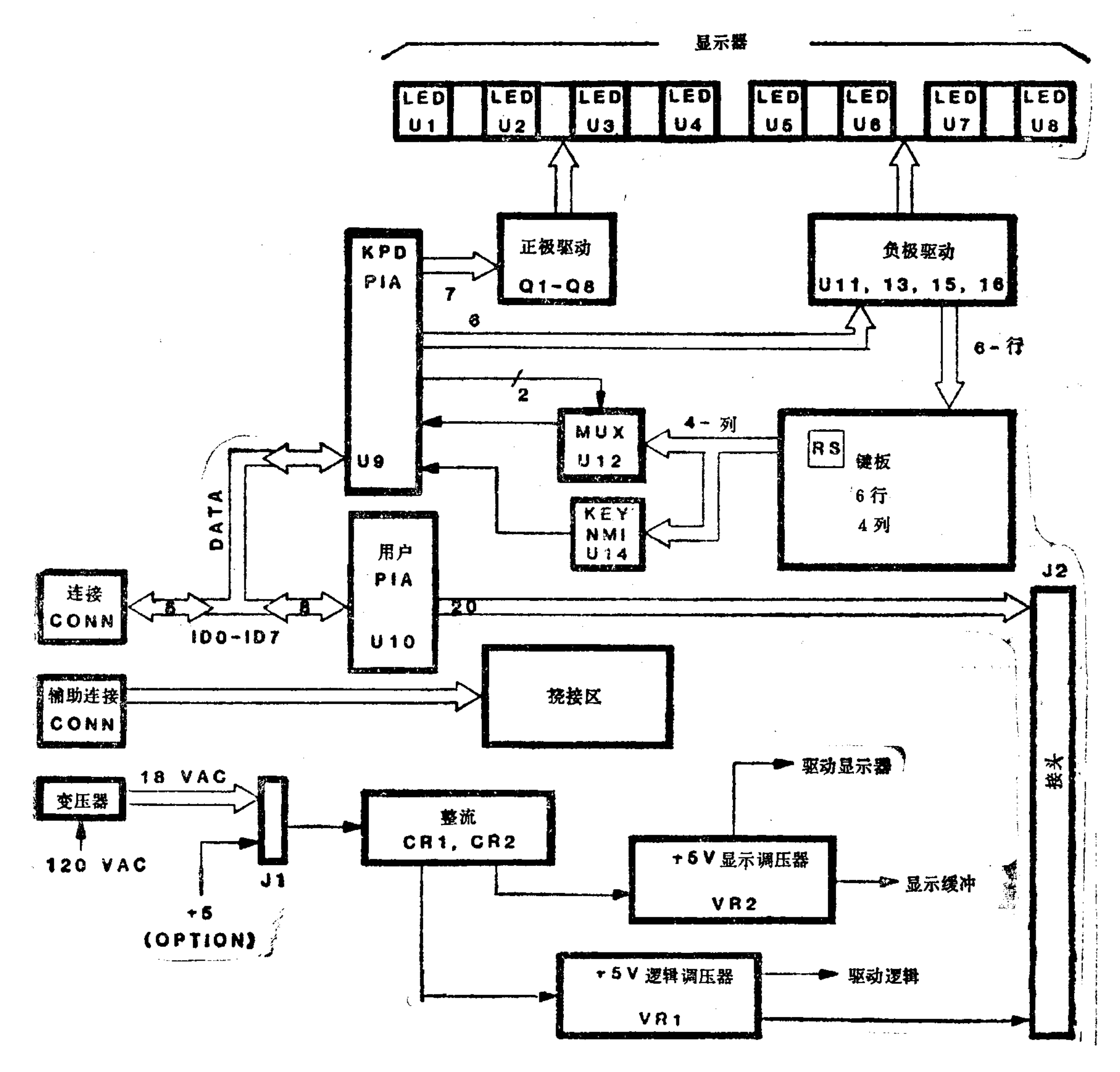
（8）盒带线路接口

D 4系统中使用很少一些元器件将盒带记录器接到其操作系统中。多数盒带操作都使用软件完成。以便把数据寄存于磁带或恢复从磁带上来的数据。磁带上的信息是用1200和2400 Hz的串行音频数据组成。

MEK6809D 4的框图和MEK68KPD的框图分别见附图12.2和附图12.3所示。



附图12.1 MEK6809D 4 框图



附图12.2 MEK68KPD框图

附录13 ASSIST 09 监控程序

1. 一般说明

6809是一种高性能的微处理器，它支持位置独立、再入和模块化程序设计技术。为了利用这些能力就需要有一个比以前所提供的监控程序在性能上更加改进和考究的用户接口。ASSIST09 监控程序充分发挥了6809的技术优越性，具有先进的性能，ASSIST09 的性能如下：

• 按位置（地址）独立方法进行程序设计，在64 K字节的地址空间的任何位置都可以执行；

• 为了装入用户改进和扩展的系统中，具有多种应用手段；

· 为了使用断点和跟踪方式开发程序，充分补充了命令；

· 为了使地址完全独立于用户服务程序，具有极考究的监控程序；

· RAM的工作区对ASSIST 0 9 ROM进行相对安排，不象其它监控程序那样在固定地址上；

· 易于适应实时工作环境；

· 可以钩链用户命令表、I/O处理程序以及省缺的技术条件；

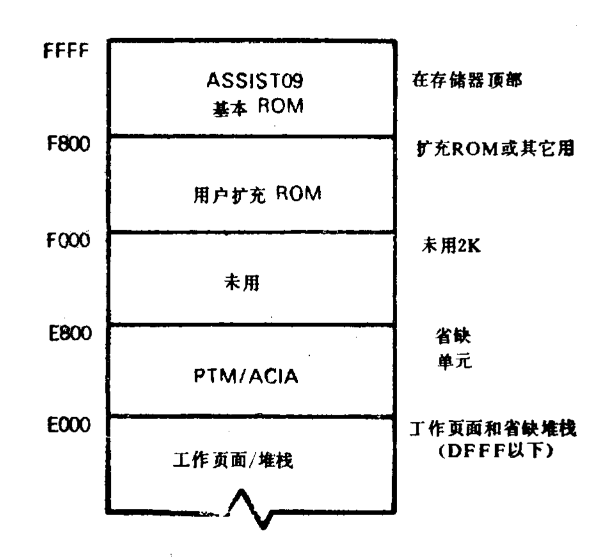
· 具有完整的标准的用户服务接口，而这些只有在完全是磁盘操作系统中才进行设置。

使用6809简明的指令系统完全能实现这些功能，而且程序量只有 2 K字节。

ASSIST 0 9 监控程序很容易在实时操作系统的控制下进行运行。该程序设有任意时间分片和强迫时间分片这些特殊功能，可以为用户程序进行几种服务程序的应用。

2． 实现技术要求

因为ASSIST09是按地址独立方法进行编码程序设计的，所以在6809的64K字节的地址空间任何地点都可正常运行。但是，为了保持各种变量和省缺的堆栈位置所需要的工作区的单元位置，还必须做一些有关的假定。该工作区是调用页面工作区，而且在ASSIST 0 9 程序内使用直接页面寄存器进行寻址。它按相对于ASSIST 0 9 ROM的起点定位，其偏值为1900_{16}。假定ASSIST0 9 放在存储器地址空间的顶部，直接控制硬件中断向量，则其存储器地址分配图将如附图13.1所示。



附图13.1 ASSIST0 9 的存储器地址分配

如果＄F800不是监控程序ROM的起点，那时的各个地址将要改变，但是，除去可编程定时器组件（PTM）和异步通信接口（ACIA）的省缺地址是固定的以外，其它程序的相对位置都保持不变。

省缺的控制台输入/输出处理程序要访问的ACIA定位在地址＄E00 8 。PTM使用的地址为 ＄E000，为了实现跟踪命令，强制产生NMI，因此可以执行单条指令。这些省缺的地址使用某些方法很容易进行改变。控制台的I/O处理程序可由用户程序重新进行安排。PTM在MONITR服务程序调用期间（见本附录第 9 节）被初始化，从而启动监控程序工作，除非

没有该省缺地址，那时也不会去访问PTM。

3. 中断控制

在总清复位基础上建立起向量表，其中包括省缺中断向量处理程序的备用地址。这些程序也可以由用户预备的向量交换服务程序（以后提到）进行重新安排。由于备用所需的省缺工作内容如下：

$\overline{RESET}$——构造ASSIST 09 向量表并建立监控程序省缺项目，然后执行监控程序中的初始化程序。

SWI——要求ASSIST0 9 服务。

$\overline{FIRQ}$——立即完成RTI。

SWI 2、SWI 3、$\overline{IRQ}$、保留项$\overline{NMI}$——实现强迫断点并进入命令处理程序。

建议在调试程序的对话过程中，使用$\overline{IRQ}$作为‘退出’功能使用，象断点和其它ASSIST0 9 省缺项利用$\overline{RESET}$进行重新初始化那样。主要使用软件中断指令SWI，并不是SWI 2 和SWI 3 指令。这样就避免了使用存储器管理单元可能会出现的错误问题，因为SWI 2和SWI3指令不禁止中断。

PTM的 1 号计数器在进行跟踪和断点命令时可以产生$\overline{NMI}$中断。在$\overline{RESET}$过程中，1号定时器的控制寄存器被初始化，以便进行跟踪。如果没进行跟踪也没设置断点，那么PTM则完全可以交给用户使用。其它情况下，只可用第 2 号和第 3 号计数器。虽然为初始化 1 号控制寄存器需要使用 2 号控制寄存器，但在$\overline{RESET}$出现之后，ASSIST09将使 2 号控制寄存器回到它原有的同一数值。所以加给用户程序的条件只是：如果 1 号控制寄存器中的“工作/予置”位必须为开启时，应该存入$A 7；如果必须为关闭时，它应该存入$A 6。

4. 初始化

在ASSIST0 9 执行期间，将使用向量表来寻址某些服务程序和省缺项数值。作出该表的目的是给用户进行修改时提供易于改变的控制信息。ASSIST 0 9 ROM的第一个字节就是一个子程序的起点，该子程序执行的任务是：初始化向量表，以及在返回调用程序之前建立起某些省缺数值。

如果ASSIST09 $\overline{RESET}$向量接收了控制，它有三项工作：

（1）在工作空间中分配指定省缺堆栈；

（2）调用上述子程序初始化向量表；

（3）启动ASSIST09监控程序，专门使用一种MONITR SWI服务请求。

然而，一个用户程序可以执行同样的功能还具有额外的好处。在调用向量初始化子程序之后，在开始正常的ASSIST09的处理之前，它可以检查或更改向量表的任何数值。这样，用户程序可以引导（或者“自举”）ASSIST09并改变省缺的标准数值。

插入用户修改项目的另一种方法是把用户程序放在ASSIST09 ROM起点之下的 2 K字节单元的扩充ROM之中。如上所述的向量表初始化程序，寻找在地址（$ 20EF）处的“BRA *”标志。而且，如果发现跟在标志后的调用单元，即为使用U寄存器指出向量表的子程序。因为是在向量表初始化之后做完这件工作的，在这时，任何一个或者所有省缺项都可以进行改变。采用这种方法的最大好处在于：当$\overline{RESET}$条件满足的基础上，修改的工作可以“自动地”进行，而改变结果即可做完，不需要外加明显的工作，就象存储器修改命令的执行一样。

在ASSIST09处理过程中没有使用专门的堆栈。这就是说，堆栈指示器必须在所有可中断的时间上都有效，而且应该有足够空间，起码有21字节的堆栈设施。在开始MONITR服务程序调用来启动ASSIST09处理期间所用的堆栈即成为“法定”堆栈。如果以后出现对该堆栈要实行有效性检查时，则在进入命令处理程序之前，这同一个堆栈要重新确定基本数值。

ASSIST09所使用的寻址工作区偏离ASSIST09 ROM起点一定的数值，该偏值为$ 1900。这一点就指出了在监控程序执行期间所使用的基准页面，而且还告知了该区包含向量表和省缺堆栈的起点。如果使用的省缺堆栈，其容量超过了81个字节，那么，在基本工作页面之下，适当扩充堆栈的邻接的RAM区必须存在。

5. 输入/输出控制

由于使用ASSIST09服务程序所产生的输出可以靠按任何电键而暂停工作，从而进入‘FREEZE’——冻结方式。再输入任意键时，就会解除这种状态，允许继续正常的输出。对于产生大量输出的那些命令可以使用CANCEL(CONTROL-X)命令令其失效。甚致在不执行控制台I/O(PAUSE服务——中止服务)时，用户程序也可以监视CANCEL和‘FREEZE’状态。

6. 命令格式

命令有三种可行的格式：

〈命令〉CR

〈命令〉〈表达式1〉CR

〈命令〉〈表达式1〉〈表达式2〉CR

间隔字符在命令和所有自变量之间作为定界符使用。设有两个专用的快速命令，对它们无需使用回车换行（CR），即“·”和“/”命令。一旦命令打错，为了重新进入该命令，需要使用CANCEL(CONTROL-X)电键。

上述的每个表达式都可由操作人员来规定的一个或多个独立的数值构成。数值内容可以是十六进制字符串、字母“P”，“M”，“W”等，也可以是某个功能的结果。每个十六进制字符串内部被转换为一个16位的二进制数。字母“P”表示使用当前程序计数器，“M”表示最后存储器检查/修改地址，“W”表示窗口数值。窗口数值的设置要使用WINDOW命令进行。

程序中还有一种INDIRECT——间接操作功能。即跟在字符“@”后的数值，使用由该数值作为地址而得到的16位数来取代。

“+”和“-”这两种运算符都可以使用，通过它们可以进行加法和减法运算。数值的运算按从左到右的顺序进行。

480——十六进制480

W+3——窗口数值加3

P-200——当前程序计数器减200_{16}

M-W——当前存储器指示器数值减窗口值

100@——为100_{16}处两字节寻址的数值

P+1@——从当前程序计数器算起采用一个字节来定位寻址的数值

7. 命令样单

附表13.1是在ASSIST09监控程序中可以使用的各种命令的样单。

附表13.1 命令样单

命令名称	说　　明	命令输入
断　点	置1、清0、显示、或删除断点	B
调　用	调用程序作为子程序	C
显　示	按十六进制和ASCII代码显示存储器数据块	D
编　码	返回变址的后缀字节数值	E
执　行	开始或继续程序执行	G
装　入	从带中装入存储器	L
存储器	检查或修改存储器	M
	最后访问的存储器修改或检查	/
	存储器修改或检查	hex/
无　效	设置新的字符和新的行号底值	N
偏　值	计算分支转移偏移值	O
记　录	存储器内容存贮到带上	P
寄存器	显示或修改寄存器内容	R
栈层次	改变堆栈跟踪层次值	S
跟　踪	跟踪指令数目	I
	跟踪一条指令	.
校　验	校验带到存储器装入内容	V
窗　口	设置窗口数值	W

8. 命令

下面将说明每条命令的使用方法，说明先后的次序是按照英文字母的顺序排列。

断点——BREAKPOINT

格式： Breakpoint

Breakpoint －

Breakpoint 〈地址〉

Breakpoint －〈地址〉

操作： 设置或改变断点表。第一种格式显示所有断点。第二种格式是清除断点表。第三种格式是把地址输入到断点表。第四种格式是从表中删除地址。在总清（RESET）时所有断点都被删除。只有在RAM中的指令才可以被断开。

调用——CALL

格式： Call

Call 〈地址〉

操作： 调用和执行作为子程序的用户程序。将要使用当前程序计数器，除非地址已被规定。用户程序应在最后结束时使用"RTS"指令。当调用出现时，将会发生断点而且程序计数器将指向监控程序。

显示——DISPLAY

格式： Display〈从〉

Display<从><长度>

Display<从><到>

操作：按十六进制和ASCII字符显示存储器的内容。当输入参数时，第二个参数如果小于第第一个参数，则表示为长度；否则当成结束地址。第一种格式是假定省缺了十六进制长度。要在模16地址字节边界之内来调整地址以包括所有字节。可以使用CANCEL（CONTROL-X）键来退出显示。在显示最后15个字节时，必须予以注意。可选项<长度>，这时总要使用，以保证正常结束：D FFE0 40

例：

D M 10——显示16字节，检查最后存储器单元。

D E000 F000——从E000到F000显示存储器内容。

编码——ENCODE

格式：Encode<变址操作数>

操作：编码命令将使变址指令寻址方式中后缀字节数值按输入的类似汇编程序语句中的操作数返回。这种命令是很有用的，特别在手编指令时。在以下例子中，表达式中所需要的16进数字使用字母H表示。

E, Y ——Y寄存器后缀字节返回到零偏值。

E[HHHH,PCR]——使用间接方式的PCR偏值退回二个字节。

E [, S++] ——返回S自动加2，间接方式。

E H, X ——X返回5位偏值方式。

注意：一个"H"规定位5位偏值，而且在偏值位置上所给数值可能是0。该命令不能查出所有规定得不正确的句法或非法变址方式。

执行——GO

格式：Go

Go<地址>

操作：从所给地址开始执行。第一种格式表示从当前程序计数器所给的内容继续执行。如果当前PC内容是一个断点，这时也不会截断，允许从断点继续执行。第二种格式中，如果规定的地址处在断点表之内，这时就按断点执行。

装入——LOAD

格式：Load

Load<偏值>

操作：装入采用S1-S9格式建立的带文件到存储器之中。如果使用偏值，则该偏值将被加到带中记录的地址之上，以便确定好实际的装入地址。所有偏值均为正值，但要注意存储器的模数为64K。由于所使用的设备不同，在装入命令完成之后，可能仍然有一些假的字符从输入设备进入并被翻译为命令字符。如果发生这种情况，应该使用CANCEL（CONTROL-X）键来消除这些字符。如果装入动作不成功，则会显示"?"。

存储器——MEMORY

格式：Memory<地址>/

<地址>/

/

操作：开始存储器检查/修改功能。第二种格式将不接受地址的表达式，只是一个十六进制字符串。第三种格式在最后存储器修改/检查功能期间，省缺显示的地址。（在表达式中使用字母“M”可得到同样数值）。

在输入了存储器命令之后，直到回车命令打入之前，仍可以使用以下动作命令：

〈Expr〉 用规定的数值代替字节，该数值可以是表达式。

SPACE （间隔）转到下一条地址并打印字节数值。

, （逗号）转到下一条地址，不打印字节数。

LF （换行）转到下一条地址，并打印该地址，同时打印字节数。

∧ （上折号或上箭头）转到前一个地址并打印该地址，同时在下一行打印字节数值。

/ 打印当前地址和在下一行的字节数。

CR （回车）结束命令。

‘〈Text〉’在第二个撇号输入之前，用ASCII字符代替以后的字节。

如果改变的意图失效（即无有效RAM），那么就会在下一个要显示的位置上出现问号。

无效——NULL

格式：Null〈说明条件〉

操作：设置新行和字符的底数数值。表征的数值被看成两个数值，高二位十六进制数表示字符底数，低二位十六进制数表示新行底数（由回车键启动）。少于三个十六进制数字的表征数将设字符底数为0。数值的范围从0到7F（十进制为127）。

例：

N 3——设字符底数为0，新行数为3。

N 207 ——设字符底数为2，新行数为7。

设置TI Silent 700终端是：

波特数	设定数
100	0
300	4
1200	317
2400	72F

偏值——OFFSET

格式：Offset〈偏值地址〉〈到的指令〉

操作：为了实现从第一表达式到第二表达式即是指令的分支转移，给出所需的一或二字节的偏值。这样，对分支转移的偏值以及变址方式指令所用的偏值都可以得到。如果给出的只是一个四字节数，那么在这两个地址之间就不能实现短分支转移。

例：

O P+2 A000——计算从当前程序计数器加2到A000所需的偏值。

记录——PUNCH

格式：Punch〈从〉〈到〉

操作：按S1-S9格式（MIKBUG监控程序所用）把二进制数记录到目标带上。

寄存器——REGISTER

格式：Register

操作：给出寄存器系列内容并提示可做改动。

在每个提示符后可以输入以下内容：

SPACE	（间隔）跳到下一个寄存器提示符。
〈Expr〉SPACE	为看下面的寄存器，而用规定的数值和提示符来代替。
〈Expr〉CR	（回车）用规定的数值和结束命令来代替。
CR	结束命令

堆栈层次——STLEVEL

格式：Stlevel

Stlevel〈地址〉

操作：设置堆栈跟踪层次，以便禁止跟踪信息。只要堆栈处在（或在）堆栈层次地址之上，就会继续跟踪显示。但当低于该地址时，即被禁止。这样就允许程序的跟踪不包括所有子程序和在跟踪信息中的低层次调用。注意一点是：通过ASSIST09"SWI"服务请求的跟踪还可以暂时限制跟踪输出，见跟踪命令说明。第一种格式是给当前程序用堆栈数值设置堆栈跟踪层次。

跟踪——TRACE

格式：Trace〈数目〉

·周期

操作：跟踪规定的指令数目。每次跟踪时，正在执行的操作码将同各个寄存器一起显示。在寄存器显示中，程序计数器显示的是要执行的下条指令。CANCEL（CONTROL-X）将提前停止跟踪。第二种格式（周期）表示单条跟踪。在跟踪期间断点没有影响。跟踪所选的部分可以使用STLEVEL命令来禁止。在ROM和RAM中的指令都可以进行跟踪，而断点只能设在RAM之中。当经由ASSIST09服务请求跟踪时，跟踪显示将被禁止，在控制返回到用户程序之前的很短时间之内，开始的两条指令进入监控程序。这样做的目的可以避免无限制地进行显示，因为有时ASSIST09为了进行请求服务，而执行大量的处理。

校验——VERIFY

格式：Verify

Verify＜偏值＞

操作：校验和比较存储器和带文件的内容。除去文件和存储器内容比较外，该命令和LOAD命令有相同格式和操作。在任何情况下如果校验有错时，都将显示一个"？"。

窗口——WINDOW

格式：窗口＜数值＞

操作：设置窗口数值。当输入字母"W"命令时，需考虑到该数值，窗口可以设置为任何一个16位数值。

9. 服务程序

下面说明由ASSIST09监控程序提供的服务程序。这些服务程序使用"SWI"指令和一个字节的功能代码来执行。所有这些服务程序在使用和操作时都可以完全按照地址独立方式工作。除非另有规定，所有寄存器在"SWI"调用期间都是透明的。在以下说明中，输入处

理程序和输出处理程序都被用到可由用户取代的备用程序之中。省缺程序通过到终端的ACIA的控制操作执行标准的I/O工作。ASCII的CANCEL代码可在大多数终端上输入，这时要同时按下CONTROL和X键。服务程序的内容如附表13.2所示：

附表13.2 服务程序样单

服务程序	输入	代码	说明
得到输入字符	INCHP	0	由输入处理程序，在寄存器A中得到输入字符
输出一个字符	OUTCH	1	从A寄存器给输出处理程序传送字符
传送字符串	PDATA1	2	给输出处理程序传送字符串
传送新行和字符串	PDATA	3	给输出处理程序传送回车、换行和字符串
转换字节为十六进制数	OUT2HS	4	按十六进制显示X寄存器表示的字节
转换字(2字节)为十六进制数	OUT4HS	5	按十六进制显示X寄存器表示的二字节字
输出到下一行	PCRLF	6	给输出处理程序传送回车、换行
传送间隔	SPACE	7	给输出处理程序传送一空白间隔
启动ASSIST09	MONITR	8	输入ASSIST09监控程序
向量交换	VCTRSW	9	检查交换向量输入表
用户断点	BRKPT	10	显示寄存器并输入命令处理程序
程序断开和检查	PAUSE	11	停止处理并检查冻结或删除状态

BRKPT——用户断点

代码：10

参数：无

结果：接受被禁止的断点，显示各寄存器内容并进入ASSIST09的命令处理程序。

说明：建立用户断点。SWI 2和SWI 3省缺备用指令同样可产生断点，但不置I和F屏蔽位。然而，因为它们两者可由用户程序代替，所以断点服务程序总可保证断点的可用性。这些用户断点不用系统断点去做，而系统断点都是由ASSIST09监控程序进行不同的处理。

例：
```
BRKPT   EQU   10      给BRKPT输入代码
        SWI           请求服务
        FCB   BRKPT   功能代码字节
```

INCHP——得到输入字符

代码：0

参数：无

结果：寄存器A从输入程序中得到字符。

说明：在从输入处理程序中接收有效输入字符之前，控制不会返回。输入字符去掉其奇偶位（第7位）并强制为0。所有的NULL（$00）和RUBOUT（$7F）字符都被忽略，而且会回到调用程序。ECHO标志，可由向量SWAP服务程序来改变，它会决定输入字符是否返回到输出处理程序（全双工操作）。在总清RESET时的省缺内容是指返回（echo）输入端。当接收回车（$0D）信号时，换行（$A0）信号自动发回给输出处理程序。

举例：
```
INCHNP     EQU     0          输入INCHP代码
SWI                           执行服务调用
FCB        INCHNP             INCHNP的功能
```
A寄存器当前是下一个字符

MONITR——启动ASSIST09

代码：8

参数：S→堆栈，变为"公务"堆栈

DP→省缺直接页面，以执行用户程序

A＝0　调用输入和输出控制初始化处理程序，并给出"ASSIST09"启动信息

A≠0　直接到命令处理程序

结果：进入ASSIST09，而且由命令处理程序进行控制。

说明：该功能实现的目的是进入ASSIST09监控程序，进入该程序可在两种情况下出现，一种是在系统总清RESET之后，另一种是在用户程序希望结束之时。如果"GO"或者"CALL"命令没做完，程序计数器在没改变之前，控制是不会返回的。ASSIST09在过去的堆栈上运行，这时如果在用户程序执行期间、堆栈出错，则该堆栈要重新确定。在应用中直接页面寄存器数值要保持省缺数值内容、以使用户程序运行。

如初始化内容所述，在调用构成子程序的向量之后，ASSIST09再启动向量程序要使用该功能来启动监控程序进行处理。如果使用A寄存器来表示，则在调用输入和输出初始化处理程序之后，要发送字符串"ASSIST09"给输出处理程序。如果可编程定时器（PTM）的地址不为0，则被初始化，所以在跟踪命令期间，可用1号寄存器产生NMI中断。然后进入命令处理程序立即执行命令请求。

举例：
```
MONITR     EQU  8             MONITR的输入代码
LOOP       CLRA               准备0页寄存器并初始化参数
*          TFR A,DP           置入省缺的页面数值
           LEAS STACK,PCR     建立省缺堆栈数值
           SWI                请求服务
           FCB MONITR         功能码字节
           BRA LOOP           如结果出现，则再进入
```

OUTCH——输出一个字符

代码：1

参数：A寄存器中内容是要发送的字节

结果：字符被送到输出处理程序。

如果发送过换行字符，字符就按以下情况置位：

如果是正常输出，CC＝0。

如果在输出中输入过CANCEL（删除），则CC＝1。

说明：如果出现冻结（接收了任意输入字符），那么在状态被解脱之前，控制本身不会返回到用户程序。只有在发出换行字符时，才会检查冻结状态。根据当前NULL（无效）命令设置的输出字符，可以发送填充性的无效字符（$00）。对于DLE（Data Link Escape——退出数据链路）的字符，绝不要发出无效字符，否则，回车换行（$00）接收新的无效行数，而所有其它字符都会成为无效字符数。

举例：
```
OUTCH   EQU    1        OUTCH的输入代码
        LDA    #'0      装入字符“0”
        SWI             用MONITOR代码发出
        FCB    OUTCH    接收代码字节
```

OUT 2 HS——字节转换为十六进制数

代码：4

参数：寄存器X指出按十六进制数显示的字节。

结果：字节本身被转换为2个十六进制数字并发给输出处理程序后跟一个空白。

举例：
```
OUT 2 HS   EQU    4            OUT 2 HS的输入代码
           LEAX DATA,PCR       给出‘DATA’去译码
           SWI                 请求服务程序
           FCB    OUT 2 HS     服务程序代码字节
```

OUT 4 HS——二字节字转换为十六进制数

代码：5

参数：寄存器X给出按十六进数字显示二字节字。

结果：二字节字被转换为4位十六进制数字，并发给输出处理程序后跟一个空白。

举例：
```
OUT 4 HS   EQU    5            OUT 4 HS的输入代码
           LEAX   DATA,PCR     装入‘DATA’地址，去译码请求
                               ASSIST09服务程序
           FCB    OUT 4 HS     服务程序代码字节
```

PAUSE——程序断开和检查

代码：11

参数：无

结果：正常返回时，CC＝0。

在中间过程如果输入了CANCEL（删除），CC＝1。

说明：在没有任何外部交互会话情况下（如控制台I/O），程序做完了所需有效的处理任务之后，都应该使用PAUSE服务程序。PAUSE服务程序另一个用处就是冻结监控或者从输入处理程序中CANCEL（删除）请求任务。这样就允许多任务的操作系统去接收控制，并可以按时间分片方式再派出其它程序。在返回之前，要执行冻结和CANCEL（删除）状态的测试。在其它任务有了执行机会之后，或者产生冻结状态之后，可以进行返回。在一个任务的系统中，只要不出现冻结状态，总要立即返回。

PCRLF——输出到下一行

代码：6

参数：无

结果：回车和换行都发送给输出处理程序。

正常输出时，C = 0。

在输出期间，输入了CONTROL-X时，CC = 1。

说明：如果出现冻结（接收了任何输入字符），那么在条件没有解脱之前，控制不会被返回到用户程序。不管是冻结状态还是CANCEL（删除）事件发生，都要完整地发出字符串。象在OUTCH服务程序中说明那样，可以发出填充性字符。

举例：
```
PCRLE    EQU    6        PCRLF的输入代码
         SWI             请求服务程序
         FCB    PCRLF    服务程序代码字节
```

PDATA——传送新行和字符串

代码：3

参数：寄存器X给出使用ASCII EOT（$04）为结尾的输出字符串。

结果：按回车和换行方式，给输出处理程序发送字符串。

正常输出时，CC = 0。

在输出期间，输入了CONTROL-X时，CC = 1

说明：输出字符串中可以包含有回车和换行字符，这样就可以使要发送的几行数据用一个功能调用。如果出现冻结(接收了任何一个输入字符)，在条件被解脱之前，控制不会返回到用户程序。不管出现什么冻结状态或是CANCEL（删除）事件，都要完整地发出字符串。象OUTCH功能一样，可以发出填充字符。

举例：
```
PDATA    EQU   3                     PDATA的输入代码
MSGOUT   FCC   'THIS IS A MULTIPLE LINE MESSAGE'。
         FCB   $OA,$OD               换行、回车
         FCC   'THIS IS THE SECOND LINE'
         FCB   $04                   字符串结束符
         LEAX  MSGOUT,PCR            装入信息地址
         SWI                         请求服务程序
         FCB   PDATA                 服务程序代码字节
```

PDATA1——发送字符串

代码：2

参数：寄存器X给出用ASCII EOT（$04）为结尾的输出字符串。

结果：字符串被发送给输出处理程序。

正常输出时，CC = 0。

在输出期间，输入了CONTROL -X时，CC = 1。

说明：输出字符串中可以包含有回车和换行字符，这样就可以使要发送的几行数据用一个功能调用。如果出现冻结（接收了任何一个输入字符），在条件被解脱之前，控制不会返回到用户程序。不管出现什么冻结状态或是CANCEL（删除）事件，都要完整地发出字符串。象OUTCH功能一样，可以发出填充字符。

```
举例：PDATA1    EQU   2                 PDATA1的输入代码
      MSG       FCC   'THIS IS AN OUTPUT STRING'
                FCB   $04               字符串结束符
                LEAX  MSG, PCR          装入'MSG'字符串地址
                SMI                     请求服务程序
                FCB   PDATA1            服务程序代码字节
```

SPACE——输出一个间隔

代码：7

参数：无

结果：一个间隔字符被送给输出处理程序。

说明：如OUTCH服务程序那样，可以发送填充字符。

```
举例：SPACE    EQU   7         SPACE的输入代码
               SWI             请求ASSIST09服务程序
               FCB   SPACE     服务程序代码字节
```

VCTRSW——向量交换

代码：9

参数：寄存器A中含有向量交换输入代码。
寄存器X中含有0或某一位移数值。

结果：寄存器X中含有上述的向量数值。

说明：向量交换服务程序检查或者修改在ASSIST09向量表中的输入字。在监控程序处理期间，该表中含有所用的指示器数值和省缺数值。只要X寄存器不是0，则输入就可以用X寄存器中含有的数值来代替。所用的各种代码如附表13.3所示。

```
举例：VCTRSW    EQU    9              VCTRSW的输入代码
      .IRQ      EQU    12             IRQ备用交换功能代码
                LEAX   MYIRQH,PCR     装入新IRQ处理程序地址
                LDA    #·IRQ          装入向量交换第二个内容
                SWI                   请求服务程序
                FCB    VCTRSW         服务程序代码字节
```

现在X是上述的备用地址

10．向量交换服务程序

向量交换程序可以使用户很容易进行对向量表的修改。每个向量处理程序都包含有一个SWI，并在任何其它处理进行之前，先在堆栈中做有效性检查。如果堆栈没有指到有效的RAM区，它就被复位到在$\overline{\text{RESET}}$之后启动ASSIST09的MONITR请求中所给出的初始数值。而且打印（显示）出当前一系列寄存器并带有问号“？”，然后进入命令处理程序。向量表中每个入口名单的内容如附表13.3所示。

附表13.3　向量表入口名单

入口名称	代码	说　　明
.AVTBL	0	向量表的返回地址
.CMDL1	2	主要命令清单
.RSVD	4	备用MC6809保留的中断向量
.SWI3	6	软件中断 3 备用中断向量
.SWI2	8	软件中断 2 备用中断向量
.FIRQ	10	备用快速中断请求向量
.IRQ	12	备用中断请求向量
.SWI	14	备用软件中断向量
.NMI	16	备用非屏蔽中断向量
.RESET	18	备用总清中断向量
.CION	20	控制台输入初始化程序
.CIDTA	22	从控制台输入数据字节程序
.CIOFF	24	输入关闭控制台程序
.COON	26	控制台输出初始化程序
.CODTA	28	给控制台输出数据字节程序
.COOFF	30	控制台关闭输出程序
.HSDTA	32	高速显示处理程序
.BSON	34	记录/装入初始化程序
.BSDTA	36	记录/装入处理程序
.BSOFF	38	记录装入关闭程序
.PAUSE	40	中止处理程序
.CMDL2	44	次要命令清单
.ACIA	46	ACIA的地址
.PAD	48	字符和新行的填充数
.ECHO	50	返回标志
.PTM	52	可编程定时器组件地址

以下来说明每个向量表入口的目的和要求，而且必须同用户取代数值或被成功代替的程序。

.ACIA——ACIA的地址

代码：46

说明：该入口包含省缺控制输入和输出设备处理程序所用的ACIA的地址。标准的ASSIST09初始化使该位数值为$E008_{16}$。如若它必须改变，那么在MONITR启动服务程序被起用之前，一定要完成，因为服务程序调用·COON和·COIN输入和输出设备初始化程序，初始化由该向量位置所给的ACIA。

AVTBL——返回向量表地址

代码：0

说明：使用该代码可返回向量表地址。不用单独地调用向量交换程序即可对向量表施行大批的改变。编码数值和向量表中的偏值相同，只要检查一下便知，该入口决不应加以改变。

.BSDTA——记录/装入处理程序

代码：36

说明：该入口包含执行记录、装入和校验操作程序的地址。在该程序做控制之前，总是先执行.BSON程序。该程序所用的参数表内容和.BSON 的相同。省缺处理程序使用.CODTA程序做记录或者按 S1/S9（MIKBUG）格式用.CIDTA程序读出数据。为了确定被处理的种类要求，必须对功能码字节进行检查。

必须给出返回代码，这部分反映了最后处理的安排：

  Z = 1　成功地完成

或Z = 0　没完成

在该程序完成之后，将调用.BSOFF程序。

.BSOFF——关闭记录/装入程序

代码：38

说明：该入口所使用的子程序目的是去结束设备本身对记录、装入和校验处理程序.BSDTA的执行。堆栈内包含的参数表作为向.BSON入口提供的资料。省缺ASSIST09程序发出DC 4（$ 14或停止）和DC 3（$ 13或断开），并跟以一秒钟延迟时间，使读出器或穿孔/记录设备有一个停止的时间。而且，由INCHP服务程序提供的内部用标志要被清除，以便抵消在.BSON处理程序中由于它的设置而带来的影响。见正确使用该标志的解释说明。

.BSON——记录/装入初始化程序

代码：34

说明：该入口所使用的子程序目的是规定在记、录、装入和校验处理中规定所用设备进行起动的任务。堆栈中含有说明所需功能的参数表。省缺程序将给输出处理程序（.CODTA）分别发出读出器启动或记录器启动的ABCII代码DC1（$ 11）或DC 2（$ 12）。同时还有标志被置位，禁止在INCHNP处理期间对冻结状态的测试。这样做是这些字符因被翻译为冻结方式指示符而并没有浪费。如果用户的取代程序还要使用的结果INCHNP服务程序，那么它还应该使这同一字节为非零，并在.BSOFF程序中清除它。ASSIST09的源程序清单应对该字节单元加以考虑。

建立的堆栈如下：

S + 6 = 代码字节，校验（- 1）、记录（0）、装入（1）

S + 4 = 只为记录用的起始地址

S + 2 = 记录的末地栈、或读出/装入的偏值

S + 0 = 返回地址

.CIDTA——从控制台输入数据字节程序

代码：22

说明：该入口确定备用控制台输入处理程序。该程序职责是给A寄存器提供所需的下一个输入字符（如果可行的话），而且根据条件码进行返回。INCHP服务程序调用该备用程序

以提供下一个字符。还有,“冻结”方式程序在不同的时间调用,以便测试冻结状态或确定冻结状态,如果CANCEL键已被输入的话。对该备用程序的处理必须遵守以下约定:

输入: PC→ASSIST09的工作页面

S→返回地址

输出: C = 0, A = 输入字符

C = 1, 如果没有输入字符可用

暂时寄存器: U, B

如果没有可用的字符,处理程序应该立即使控制返回。这样在输入无效时,可使其它任务去完成大量工作。省缺程序读出ACIA的问题见本附录第2节实现技术要求中的说明。

.CIOFF——输入关闭控制台程序

代码: 24

说明: 该入口给出结束输入处理时所调用的程序。在任何时间,它不用ASSIST09调用, 但其包含有一致性的内容。省缺程序仅做“RTS”。其环境如下:

输入: 无

输出: 输入设备结束

暂时性寄存器: 无

.CION——输入初始化控制台程序

代码: 20

说明: 该入口在初始化输入设备时进行调用。在MONITR服务程序中一旦它被调用时, 它就会初始化监控程序,于是命令处理程序可以得到命令进行处理。省缺处理程序总清复位为标准输入和输出用的ACIA, 并建立以下省缺条件: 8位字长、无奇偶检验、2个终止位、计数器除以16。一个无奇偶位校验的8位字的作用是接收7位ASCII编码并忽略奇偶位。

输入: .ACIA的ACIA的存储器地址

输出: 输出设备被初始化

暂时性寄存器: A, X

.CMDL1——主要命令清单

代码: 2

说明: 用户提供的命令表可以代替或取代ASSIST09的标准表。命令处理程序扫描两个清单,主表在前,次表在后。主表由该入口给出,并包含ASSIST09命令表,好象省缺项。次表省缺了一个无效清单。用可以把自己的表插入两个表中任何一个表之内。如果用户清单被安排在次表之中,那么ASSIST09的清单将首先被检索。省缺ASSIST09清单包含一个字符的所有命令名称。所以用户命令“PRINT”应和键入“PR”的命令一样,但不能只是键入“P”。因为系统命令清单首先要内容上匹配。如果需要, 用户可以取代主系统清单。命令被选的基础首先是同输入字符进行比较,这就是说,两条或更多条的命令之间的开始字符可能是相同的,所以只有输入多个字符时,才可在清单中首先被选。在用户命令清单中的每一入口,必须有以下格式:

+0	FCB	L	其中“L”是包括该字节在内的入口尺寸
+1	FCC	‘<字符串>’	其中“<字符串>”是命令名称

+N　　FDB　　EP-*　　其中“EP”代表定义命令程序开始的符号

第一个字节是入口长度字节，而且总是有三个字节，并大于命令字符串的长度（长度本身为一个字节，程序偏值为二个字节）。命令字符串必须只为ASCII字符，没有特殊字符，命令程序的起始的偏值被用以代替绝对地址，所以位置独立程序可以包括命令表在内。命令表的尾部是一个字节的标志。-1（$FF）表示进行检索的是次表；-2（$FE）表示要结束命令表的检索。次命令清单所表示的命令表必须用2结束。如果两个清单都被检索，第一个清单必须用-1结束；如果只用一个清单，则第一个清单必须用-2结束。

进入命令程序用到下面的各个寄存器：

DPR→　ASSIST09页面工作区

S→　命令处理程序的返回地址

Z=1　命令名称结束时的回车

Z=0　跟随命令名称的间隔定义符

在键入命令名称定义符之后，进入命令程序。这就是说，回车可以是由停止在下一行的输入设备所输入的定义符。因此，条件码中的Z位被置1，命令程序即可确定输入设备的当前位置。命令程序应该保证，在回到命令处理程序之前，控制台设备应被放在新的一行之上。

.CMDL2——次命令清单

代码：44

说明：该入口给出第二个清单表。省缺内容是一无效清单，后跟一个-2字节。对该入口使用的完整说明，在.CMDL 1入口的说明中作过介绍。

.CODTA——给控制台输出数据字节的程序

代码：28

说明：该程序的职责是把A寄存器中的字符发送给输出设备。省缺程序也带有填充性字符，如在OUTCH服务程序中说明的一样。如果输出设备为接收字符还没准备就绪，那么在这种条件最终成立之前，应重复调用“中止”子程序。该子程序的地址将从向量表中.PAUSE入口中取得。填充的字符数从向量表中.PAD入口中得到。ASSIST09的所有输出都是调用这种备用程序完成的。这种备用程序也含有穿孔或记录处理过程。省缺程序给ACIA发送字符的过程，已在本附录第2节实现技术要求中做过说明。操作环境如下：

输入：A=发送字符

　　DP=ASSIST09工作页面

　　.PAD=字符和新行的填充数（在向量表中）

　　.PAUSE=中止程序（向量表中）

输出：给输出设备发送字符

暂时性寄存器：无。所有工作寄存器都须恢复。

.COOFF——控制台关闭输出程序

代码：30

说明：该入口的程序可以结束输出设备的处理。ASSIST09不调用该程序。为了完整性起见，

它被包含在其中。省缺程序是“RTS”。

输入：DP→ASSIST09工作页面

输出：结束输出设备的处理

暂时性寄存器：无

.COON——控制台输出初始化程序

代码：26

说明：该入口所指程序是初始化标准的输出设备。省缺程序初始化ACIA，与下面.CION向量交换定义中所说明的内容非常相象。

输入：给ACIA地址用的.ACIA向量入口

输出：初始化输出设备

暂时性寄存器：A,X

.ECHO——返回标志

代码：50

说明：该字的第一个字节作为INCHP服务程序的标志，以便确定从输入处理程序所接收的返回输入的要求。数值不为0时表示要返回输入端；数值为0时不要返回输入端。即使用户程序被省缺.CIDTA处理程序取代，返回作用仍将存在，与INCHP服务程序执行返回工作一样。

.FIRQ——备用快速中断请求向量

代码：10

说明：快速中断请求程序经该指示器进行分配。MC6809在处理$\overline{FIRQ}$时所确定的处理程序向量地址为FFF6。在该备用入口上为$\overline{FIRQ}$中断规定了堆栈和机器的状态。这里要说明的是采用间接跳转指令“跳越转移”到该程序的，所以在处理程序实际接受控制之前，对中断时间来说增加了11个机器周期。省缺程序立即做“RTI”指令，实际上是忽略了中断。

.HSDTA——高速显示处理程序

代码：32

说明：引入该程序的目的是作为显示——DISPLAY命令的子程序使用，并且传送有“到”和“从”地址在内的参数表。这个从地址的边界可下至16个字节的地址。省缺程序按十六进制数值和ASCII编码两种方式来显示存储器内容，而且每经128字节即产生一个台头标题。采用该向量表入口的目的是使专门处理数据块的用户程序易于实现（譬如，使数据可以快速地送到高速打印机，以利后来的分析）。各参数全被送到堆栈。其工作环境如下：

输入：S + 4 =起始地址

S + 2 =结束地址

S + 0 =返回地址

DP→ASSIST09工作页面

输出：任意要求

暂时性寄存器：X，D

.IRQ——备用中断请求向量

代码：12

说明：所有中断请求都被送到由该向量给出的程序。MC6809的中断向量取出单元地址是FFF 8。为该处理程序的入口上的$\overline{IRQ}$中断定义了堆栈和处理机的状态。因为程序的地址在向量表之中。所以为了引用该程序一定要做间接跳转程序。在$\overline{IRQ}$处理程序接受控制之前的中断处理时间增加了11个机器周期。省缺$\overline{IRQ}$处理程序打印各寄存器内容，并进入ASSIST09的命令处理程序。

.NMI——备用非屏蔽中断向量

代码：16

说明：不管处理机何时转移到FFFC的地址，该入口都会指派给非屏蔽中断处理程序，以便接受控制。因为在跟踪和断点处理期间ASSIST09使用$\overline{NMI}$中断，所以如果用户处理程序进行控制时，不应该使用该命令。如果$\overline{NMI}$中断要由用户设备产生，那么就需要使用户处理程序了解并可以做到将控制转送给省缺处理程序，否则是不行的。$\overline{NMI}$处理程序在控制上增加了11个周期的时间开销，因为其地址要从向量表中取出。

.PAD——字符和新行的填充数

代码：48

说明：该入口包含有字符和新行的填充数。前一个字节是其它字符所用的无效（$00）数，第二个字节是在某个时间发送了换行字符之后而发出的无效（$00）数。在ASCII退出字符（$10）之后决不会发出无效数（$00）。省缺程序处理.CODTA的任务是发送这些无效数。需要时用户处理程序可以使用这些数，也可以不用。

使用“NULLS”命令还可以使这两个字节设置为用户所规定的数值。

.PAUSE——中止处理程序

代码：40

说明：为了支持实时环境（即多任务要求），ASSIST09可调用一个无效时间程序，随时可以使处理过程来等待某些外部状态的变化。有一个例子可以很好说明该问题，这就是在OUTCH服务程序中，当经过省缺处理程序.CODTA给ACIA发送字符时，ACIA的状态寄存器内容表明它还不能接收。省缺无效时间程序放在了一个4字节的保留区，其中包含一条指令“RTS”。.PAUSE向量入口在正常初始化之后就指向该程序。该指示器可以进行改变，使其指向用户程序，进而指派其它程序，所以对MC6809可以更充分地利用。另一个例子就是计数器增值计数，所以无效时间周期数，可以为统计目的或调试目的而进行累加。设四个字节保留区的理由（它处在ASSIST09的工作页面中）是其它的代码可以进行覆盖，而无需指派地址分配图中另外的空间。例如，主监控程序可以使用存储器管理单元给ASSIST09分配整个64K存储器空间，并在ASSIST09的控制下，使程序执行或进行测试。当然，在任何“无效时间”出现之时，主监控程序都希望进行再入，所以它要使用它自己的“SWI”来覆盖省缺程序（“RTS”）。因为主监控程序无论如何要“提前结束”所有的“SWI”，所以它知道当“中止”调用在执行时，可以在时间分片的基础上重新调度其它系统。

通过中止处理程序的所有寄存器一定都是透明的。同在ASSIST09用户服务程序处理中的被选点一起，为用户程序有一个特殊的专门服务调用以便利用中止程序。通常可建议，如果在所给时间周期之内（比如10ms），没有服务程序请求，那么，用户程

序应调用.PAUSE服务程序，所以，可以保证中等任务的派遣。

.PTM——可编程定时器组件地址

代码： 53

说明： 该入口含有MC6840程控定时器组件（PTM）的地址。在第4节初始化中说明过，在MONITR启动服务程序被调用之前，该组件的位置变动应该完毕。如果不用PTM，则地址被改变为0，所以对其不存在初始化。需要注意的是：如果设置为0，那么，ASSIST09就不应该发出断点和跟踪命令。

.RESET——备用RESET总清中断向量

代码： 18

说明： 该入口要返回到初始化ASSIST09的RESET程序的地址，对该入口的改变没有影响，但被包含在向量表之中，在这种情况下，用户程序希望确定ASSIST09的再启动的所在位置。例如，如果ASSIST09处在存储器地址分配图之中，但是它不控制MC6809的硬件向量，那么，用户程序要启动它时，就需要取得正常的RESET向量代码的地址。ASSIST09的总清代码指定为工作页面中的省缺内容，并调用构造子程序的向量，然后使用MONITR服务程序调用正常地启动ASSIST09。

.RSVD——MC6809备用保留的中断向量

代码： 4

说明： 该入口是一个地址为$FFF0_{16}$的被保留的中断向量程序使用的指示字。MC6809的硬件向量现在还没有进行定义。由ASSIST09建立的省缺程序可以执行寄存器显示，并进入命令处理程序。

.SWI——备用软件中断向量

代码： 14

说明： 该向量入口含有软件中断程序的地址。正常情况下，ASSIST09处理这些中断来给用户程序提供服务。但如果用户处理程序在位，就不能使用这些措施，否则，用户程序就需把这种请求继续传递给ASSIST09的省缺处理程序。这一点是很容易实现的，因为当交换由用户能够完成时，向量交换功能可以送回省缺处理程序的地址。“提前结束”的做法可以使用户程序对所有服务程序的调用进行检查，或者改变/取代/扩充对这些用户程序的要求。当然，所有的寄存器在控制从用户传送给标准的处理程序中都必须是透明的。当SMI出现时，“JMP”指令可以直接转移到该向量入口所指派的程序。所以工作的环境条件在于对“SWI”中断作出规定。

.SWI 2——备用软件中断2向量

代码： 8

说明： 该入口包含一个SWI 2处理程序的指示字，在执行指令的任何时候都可以对它进行输入。有关的这些堆栈和机器的状态都要由SWI 2的中断来定义，该中断的向量地址在FFF 4。省缺处理程序的任务是打印或显示各寄存器内容并进入ASSIST09的命令处理程序。

.SWI 3——备用软件中断3向量

代码： 6

说明： 该入口包含一个SWI 3处理程序的指示字，在执行指令的任何时候都可以对它进行输

入。有关的这些堆栈和机器的状态都要由SWI 3 的中断来定义，该中断的向量地址安排在 FFF 2 。省缺处理程序的任务是打印或显示各寄存器内容并进 入 ASSIST09 的命令处理程序。

附表13.4 ASSIST09 监控程序清单

```
PAGE  001  ASSIST09.SA:0          ASSIST09 - MC6809 MONITOR

00001                              TTL     ASSIST09 - MC6809 MONITOR
00002                              OPT     ABS,LLE=85,S,CRE

00004                       ***************************************
00005                       * COPYRIGHT (C) MOTOROLA, INC. 1979 *
00006                       ***************************************

00008                       ***************************************
00009                       *  THIS IS THE BASE ASSIST09 ROM.
00010                       *  IT MAY RUN WITH OR WITHOUT THE
00011                       *  EXTENSION ROM WHICH
00012                       *  WHEN PRESENT WILL BE AUTOMATICALLY
00013                       *  INCORPORATED BY THE BLDVTR
00014                       *  SUBROUTINE.
00015                       ***************************************

00017                       ***********************************************
00018                       *             GLOBAL MODULE EQUATES
00019                       ***********************************************
00020           F800     A ROMBEG EQU     $F800     ROM START ASSEMBLY ADDRESS
00021           E700     A RAMOFS EQU     -$1900    ROM OFFSET TO RAM WORK PAGE
00022           0800     A ROMSIZ EQU     2048      ROM SIZE
00023           F000     A ROM2OF EQU     ROMBEG-ROMSIZ START OF EXTENSION ROM
00024           E008     A ACIA   EQU     $E008     DEFAULT ACIA ADDRESS
00025           E000     A PTM    EQU     $E000     DEFAULT PTM ADDRESS
00026           0000     A DFTCHP EQU     0         DEFAULT CHARACTER PAD COUNT
00027           0005     A DFTNLP EQU     5         DEFAULT NEW LINE PAD COUNT
00028           003E     A PROMPT EQU     '>        PROMPT CHARACTER
00029           0008     A NUMBKP EQU     8         NUMBER OF BREAKPOINTS
00030                       ***********************************************

00032                       ***********************************************
00033                       *  MISCELANEOUS EQUATES
00034                       ***********************************************
00035           0004     A EOT    EQU     $04       END OF TRANSMISSION
00036           0007     A BELL   EQU     $07       BELL CHARACTER
00037           000A     A LF     EQU     $0A       LINE FEED
00038           000D     A CR     EQU     $0D       CARRIAGE RETURN
00039           0010     A DLE    EQU     $10       DATA LINK ESCAPE
00040           0018     A CAN    EQU     $18       CANCEL (CTL-X)
00041                       * PTM ACCESS DEFINITIONS
00042           E001     A PTMSTA EQU     PTM+1     READ STATUS REGISTER
00043           E000     A PTMC13 EQU     PTM       CONTROL REGISTERS 1 AND 3
00044           E001     A PTMC2  EQU     PTM+1     CONTROL REGISTER 2
00045           E002     A PTMTM1 EQU     PTM+2     LATCH 1
00046           E004     A PTMTM2 EQU     PTM+4     LATCH 2
00047           E006     A PTMTM3 EQU     PTM+6     LATCH 3

00049           008C     A SKIP2  EQU     $8C       "CMPX #" OPCODE - SKIPS TWO BYTES

00051                       ***********************************************
00052                       *     ASSIST09 MONITOR SWI FUNCTIONS
```

```
00053                          * THE FOLLOWING EQUATES DEFINE FUNCTIONS PROVIDED
00054                          * BY THE ASSIST09 MONITOR VIA THE SWI INSTRUCTION.
00055                          ******************************************
00056         0000     A INCHNP  EQU    0         INPUT CHAR IN A REG - NO PARITY
00057         0001     A OUTCH   EQU    1         OUTPUT CHAR FROM A REG
00058         0002     A PDATA1  EQU    2         OUTPUT STRING
00059         0003     A PDATA   EQU    3         OUTPUT CR/LF THEN STRING
00060         0004     A OUT2HS  EQU    4         OUTPUT TWO HEX AND SPACE
00061         0005     A OUT4HS  EQU    5         OUTPUT FOUR HEX AND SPACE
00062         0006     A PCRLF   EQU    6         OUTPUT CR/LF
00063         0007     A SPACE   EQU    7         OUTPUT A SPACE
00064         0008     A MONITR  EQU    8         ENTER ASSIST09 MONITOR
00065         0009     A VCTRSW  EQU    9         VECTOR EXAMINE/SWITCH
00066         000A     A BRKPT   EQU    10        USER PROGRAM BREAKPOINT
00067         000B     A PAUSE   EQU    11        TASK PAUSE FUNCTION
00068         000B     A NUMFUN  EQU    11        NUMBER OF AVAILABLE FUNCTIONS
00069                          * NEXT SUB-CODES FOR ACCESSING THE VECTOR TABLE.
00070                          * THEY ARE EQUIVALENT TO OFFSETS IN THE TABLE.
00071                          * RELATIVE POSITIONING MUST BE MAINTAINED.
00072         0000     A .AVTBL  EQU    0         ADDRESS OF VECTOR TABLE
00073         0002     A .CMDL1  EQU    2         FIRST COMMAND LIST
00074         0004     A .RSVD   EQU    4         RESERVED HARDWARE VECTOR
00075         0006     A .SWI3   EQU    6         SWI3 ROUTINE
00076         0008     A .SWI2   EQU    8         SWI2 ROUTINE
00077         000A     A .FIRQ   EQU    10        FIRQ ROUTINE
00078         000C     A .IRQ    EQU    12        IRQ ROUTINE
00079         000E     A .SWI    EQU    14        SWI ROUTINE
00080         0010     A .NMI    EQU    16        NMI ROUTINE
00081         0012     A .RESET  EQU    18        RESET ROUTINE
00082         0014     A .CION   EQU    20        CONSOLE ON
00083         0016     A .CIDTA  EQU    22        CONSOLE INPUT DATA
00084         0018     A .CIOFF  EQU    24        CONSOLE INPUT OFF
00085         001A     A .COON   EQU    26        CONSOLE OUTPUT ON
00086         001C     A .CODTA  EQU    28        CONSOLE OUTPUT DATA
00087         001E     A .COOFF  EQU    30        CONSOLE OUTPUT OFF
00088         0020     A .HSDTA  EQU    32        HIGH SPEED PRINTDATA
00089         0022     A .BSON   EQU    34        PUNCH/LOAD ON
00090         0024     A .BSDTA  EQU    36        PUNCH/LOAD DATA
00091         0026     A .BSOFF  EQU    38        PUNCH/LOAD OFF
00092         0028     A .PAUSE  EQU    40        TASK PAUSE ROUTINE
00093         002A     A .EXPAN  EQU    42        EXPRESSION ANALYZER
00094         002C     A .CMDL2  EQU    44        SECOND COMMAND LIST
00095         002E     A .ACIA   EQU    46        ACIA ADDRESS
00096         0030     A .PAD    EQU    48        CHARACTER PAD AND NEW LINE PAD
00097         0032     A .ECHO   EQU    50        ECHO/LOAD AND NULL BKPT FLAG
00098         0034     A .PTM    EQU    52        PTM ADDRESS
00099         001B     A NUMVTR  EQU    52/2+1    NUMBER OF VECTORS
00100         0034     A HIVTR   EQU    52        HIGHEST VECTOR OFFSET
```

```
00102                         ******************************************
00103                         *                  WORK AREA
00104                         * THIS WORK AREA IS ASSIGNED TO THE PAGE ADDRESSED BY
00105                         * -$1800,PCR FROM THE BASE ADDRESS OF THE ASSIST09
00106                         * ROM.  THE DIRECT PAGE REGISTER DURING MOST ROUTINE
00107                         * OPERATIONS WILL POINT TO THIS WORK AREA.  THE STACK
00108                         * INITIALLY STARTS UNDER THE RESERVED WORK AREAS AS
00109                         * DEFINED HEREIN.
00110                         ******************************************
00111        DF00    A WORKPG EQU    ROMBEG+RAMOFS SETUP DIRECT PAGE ADDRESS
00112        00DF    A        SETDP  WORKPG!>8 NOTIFY ASSEMBLER
00113A E000                   ORG    WORKPG+256 READY PAGE DEFINITIONS
00114                         * THE FOLLOWING THRU BKPTOP MUST RESIDE IN THIS ORDER
00115                         * FOR PROPER INITIALIZATION
00116A DFFC                   ORG    *-4
00117        DFFC    A PAUSER EQU    *        PAUSE ROUTINE
00118A DFFB                   ORG    *-1
00119        DFFB    A SWIBFL EQU    *        BYPASS SWI AS BREAKPOINT FLAG
00120A DFFA                   ORG    *-1
00121        DFFA    A BKPTCT EQU    *        BREAKPOINT COUNT
00122A DFF8                   ORG    *-2
00123        DFF8    A SLEVEL EQU    *        STACK TRACE LEVEL
00124A DFC2                   ORG    *-NUMVTR*2
00125        DFC2    A VECTAB EQU    *        VECTOR TABLE
00126A DFB2                   ORG    *-2*NUMBKP
00127        DFB2    A BKPTBL EQU    *        BREAKPOINT TABLE
00128A DFA2                   ORG    *-2*NUMBKP
00129        DFA2    A BKPTOP EQU    *        BREAKPOINT OPCODE TABLE
00130A DFA0                   ORG    *-2
00131        DFA0    A WINDOW EQU    *        WINDOW
00132A DF9E                   ORG    *-2
00133        DF9E    A ADDR   EQU    *        ADDRESS POINTER VALUE
00134A DF9D                   ORG    *-1
00135        DF9D    A BASEPG EQU    *        BASE PAGE VALUE
00136A DF9B                   ORG    *-2
00137        DF9B    A NUMBER EQU    *        BINARY BUILD AREA
00138A DF99                   ORG    *-2
00139        DF99    A LASTOP EQU    *        LAST OPCODE TRACED
00140A DF97                   ORG    *-2
00141        DF97    A RSTACK EQU    *        RESET STACK POINTER
00142A DF95                   ORG    *-2
00143        DF95    A PSTACK EQU    *        COMMAND RECOVERY STACK
00144A DF93                   ORG    *-2
00145        DF93    A PCNTER EQU    *        LAST PROGRAM COUNTER
00146A DF91                   ORG    *-2
00147        DF91    A TRACEC EQU    *        TRACE COUNT
00148A DF90                   ORG    *-1
00149        DF90    A SWICNT EQU    *        TRACE "SWI" NEST LEVEL COUNT
00150A DF8F                   ORG    *-1      (MISFLG MUST FOLLOW SWICNT)
00151        DF8F    A MISFLG EQU    *        LOAD CMD/THRU BREAKPOINT FLAG
00152A DF8E                   ORG    *-1
00153        DF8E    A DELIM  EQU    *        EXPRESSION DELIMITER/WORK BYTE
00154A DF66                   ORG    *-40
00155        DF66    A ROM2WK EQU    *        EXTENSION ROM RESERVED AREA
00156A DF51                   ORG    *-21
00157        DF51    A TSTACK EQU    *        TEMPORARY STACK HOLD
00158        DF51    A STACK  EQU    *        START OF INITIAL STACK
```

```
00160                          **********************************************
00161                          * DEFAULT THE ROM BEGINNING ADDRESS TO 'ROMBEG'
00162                          * ASSIST09 IS POSITION ADDRESS INDEPENDENT, HOWEVER
00163                          * WE ASSEMBLE ASSUMING CONTROL OF THE HARDWARE VECTORS.
00164                          * NOTE THAT THE WORK RAM PAGE MUST BE 'RAMOFS'
00165                          * FROM THE ROM BEGINNING ADDRESS.
00166                          ***********************************************
00167A F800                          ORG    ROMBEG   ROM ASSEMBLY/DEFAULT ADDRESS

00169                          *****************************************************
00170                          *               BLDVTR - BUILD ASSIST09 VECTOR TABLE
00171                          *  HARDWARE RESET CALLS THIS SUBROUTINE TO BUILD THE
00172                          *  ASSIST09 VECTOR TABLE.  THIS SUBROUTINE RESIDES AT
00173                          *  THE FIRST BYTE OF THE ASSIST09 ROM, AND CAN BE
00174                          *  CALLED VIA EXTERNAL CONTROL CODE FOR REMOTE
00175                          *  ASSIST09 EXECUTION.
00176                          * INPUT: S->VALID STACK RAM
00177                          * OUTPUT: U->VECTOR TABLE ADDRESS
00178                          *         DPR->ASSIST09 WORK AREA PAGE
00179                          *         THE VECTOR TABLE AND DEFAULTS ARE INITIALIZED
00180                          *  ALL REGISTERS VOLATILE
00181                          *****************************************************

00183A F800 30   8D E7BE   BLDVTR LEAX   VECTAB,PCR ADDRESS VECTOR TABLE
00184A F804 1F   10      A        TFR    X,D      OBTAIN BASE PAGE ADDRESS
00185A F806 1F   8B      A        TFR    A,DP     SETUP DPR
00186A F808 97   9D      A        STA    BASEPG   STORE FOR QUICK REFERENCE
00187A F80A 33   84      A        LEAU   ,X       RETURN TABLE TO CALLER
00188A F80C 31   8C 35            LEAY   <INITVT,PCR LOAD FROM ADDR
00189A F80F EF   81      A        STU    ,X++     INIT VECTOR TABLE ADDRESS
00190A F811 C6   16      A        LDB    #NUMVTR-5 NUMBER RELOCATABLE VECTORS
00191A F813 34   04      A        PSHS   B        STORE INDEX ON STACK
00192A F815 1F   20      A BLD2   TFR    Y,D      PREPARE ADDRESS RESOLVE
00193A F817 E3   A1      A        ADDD   ,Y++     TO ABSOLUTE ADDRESS
00194A F819 ED   81      A        STD    ,X++     INTO VECTOR TABLE
00195A F81B 6A   E4      A        DEC    ,S       COUNT DOWN
00196A F81D 26   F6   F815        BNE    BLD2     BRANCH IF MORE TO INSERT
00197A F81F C6   0D      A        LDB    #INTVE-INTVS STATIC VALUE INIT LENGTH
00198A F821 A6   A0      A BLD3   LDA    ,Y+      LOAD NEXT BYTE
00199A F823 A7   80      A        STA    ,X+      STORE INTO POSITION
00200A F825 5A                    DECB            COUNT DOWN
00201A F826 26   F9   F821        BNE    BLD3     LOOP UNTIL DONE
00202A F828 31   8D F7D4          LEAY   ROM2OF,PCR TEST POSSIBLE EXTENSION ROM
00203A F82C 8E   20FE    A        LDX    #$20FE   LOAD "BRA *" FLAG PATTERN
00204A F82F AC   A1      A        CMPX   ,Y++     ? EXTENDED ROM HERE
00205A F831 26   02   F835        BNE    BLDRTN   BRANCH NOT OUR ROM TO RETURN
00206A F833 AD   A4      A        JSR    ,Y       CALL EXTENDED ROM INITIALIZE
00207A F835 35   84      A BLDRTN PULS   PC,B     RETURN TO INITIALIZER

00209                          *****************************************************
00210                          *                  RESET ENTRY POINT
00211                          *  HARDWARE RESET ENTERS HERE IF ASSIST09 IS ENABLED
00212                          *  TO RECEIVE THE MC6809 HARDWARE VECTORS.  WE CALL
00213                          *  THE BLDVTR SUBROUTINE TO INITIALIZE THE VECTOR
```

```
00214                          *  TABLE, STACK, AND THEN FIREUP THE MONITOR VIA SWI
00215                          *  CALL.
00216                          *****************************************************
00217A F837 32   8D E716       RESET  LEAS   STACK,PCR SETUP INITIAL STACK
00218A F83B 8D   C3   F800            BSR    BLDVTR    BUILD VECTOR TABLE
00219A F83D 4F                 RESET2 CLRA             ISSUE STARTUP MESSAGE
00220A F83E 1F   8B      A            TFR    A,DP      DEFAULT TO PAGE ZERO
00221A F840 3F                        SWI              PERFORM MONITOR FIREUP
00222A F841      08      A            FCB    MONITR    TO ENTER COMMAND PROCESSING
00223A F842 20   F9   F83D            BRA    RESET2    REENTER MONITOR IF 'CONTINUE'

00225                          ****************************************************
00226                          *          INITVT - INITIAL VECTOR TABLE
00227                          *  THIS TABLE IS RELOCATED TO RAM AND REPRESENTS THE
00228                          *  INITIAL STATE OF THE VECTOR TABLE. ALL ADDRESSES
00229                          *  ARE CONVERTED TO ABSOLUTE FORM.  THIS TABLE STARTS
00230                          *  WITH THE SECOND ENTRY, ENDS WITH STATIC CONSTANT
00231                          *  INITIALIZATION DATA WHICH CARRIES BEYOND THE TABLE.
00232                          **************************************************
00233A F844      0158    A INITVT FDB    CMDTBL-* DEFAULT FIRST COMMAND TABLE
00234A F846      0292    A        FDB    RSRVDR-* DEFAULT UNDEFINED HARDWARE VECTOR
00235A F848      0290    A        FDB    SWI3R-*  DEFAULT SWI3
00236A F84A      028E    A        FDB    SWI2R-*  DEFAULT SWI2
00237A F84C      0270    A        FDB    FIRQR-*  DEFAULT FIRQ
00238A F84E      028A    A        FDB    IRQR-*   DEFAULT IRQ ROUTINE
00239A F850      0045    A        FDB    SWIR-*   DEFAULT SWI ROUTINE
00240A F852      022B    A        FDB    NMIR-*   DEFAULT NMI ROUTINE
00241A F854      FFE3    A        FDB    RESET-*  RESTART VECTOR
00242A F856      0290    A        FDB    CION-*   DEFAULT CION
00243A F858      0284    A        FDB    CIDTA-*  DEFAULT CIDTA
00244A F85A      0296    A        FDB    CIOFF-*  DEFAULT CIOFF
00245A F85C      028A    A        FDB    COON-*   DEFAULT COON
00246A F85E      0293    A        FDB    CODTA-*  DEFAULT CODTA
00247A F860      0290    A        FDB    COOFF-*  DEFAULT COOFF
00248A F862      039A    A        FDB    HSDTA-*  DEFAULT HSDTA
00249A F864      02B7    A        FDB    BSON-*   DEFAULT BSON
00250A F866      02D2    A        FDB    BSDTA-*  DEFAULT BSDTA
00251A F868      02BF    A        FDB    BSOFF-*  DEFAULT BSOFF
00252A F86A      E792    A        FDB    PAUSER-* DEFAULT PAUSE ROUTINE
00253A F86C      047D    A        FDB    EXP1-*   DEFAULT EXPRESSION ANALYZER
00254A F86E      012D    A        FDB    CMDTB2-* DEFAULT SECOND COMMAND TABLE
00255                    * CONSTANTS
00256A F870      E008    A INTVS  FDB    ACIA     DEFAULT ACIA
00257A F872      00      A        FCB    DFTCHP,DFTNLP DEFAULT NULL PADDS
00258A F874      0000    A        FDB    0        DEFAULT ECHO
00259A F876      E000    A        FDB    PTM      DEFAULT PTM
00260A F878      0000    A        FDB    0        INITIAL STACK TRACE LEVEL
00261A F87A      00      A        FCB    0        INITIAL BREAKPOINT COUNT
00262A F87B      00      A        FCB    0        SWI BREAKPOINT LEVEL
00263A F87C      39      A        FCB    $39      DEFAULT PAUSE ROUTINE (RTS)
00264            F87D    A INTVE  EQU    *
00265                    *B

00267                    ***************************************************
```

```
00268                         *             ASSIST09 SWI HANDLER
00269                         * THE SWI HANDLER PROVIDES ALL INTERFACING NECESSARY
00270                         * FOR A USER PROGRAM.  A FUNCTION BYTE IS ASSUMED TO
00271                         * FOLLOW THE SWI INSTRUCTION.  IT IS BOUND CHECKED
00272                         * AND THE PROPER ROUTINE IS GIVEN CONTROL.  THIS
00273                         * INVOCATION MAY ALSO BE A BREAKPOINT INTERRUPT.
00274                         * IF SO, THE BREAKPOINT HANDLER IS ENTERED.
00275                         * INPUT: MACHINE STATE DEFINED FOR SWI
00276                         * OUTPUT: VARIES ACCORDING TO FUNCTION CALLED. PC ON
00277                         *     CALLERS STACK INCREMENTED BY ONE IF VALID CALL.
00278                         * VOLATILE REGISTERS: SEE FUNCTIONS CALLED
00279                         * STATE: RUNS DISABLED UNLESS FUNCTION CLEARS I FLAG.
00280                         ***************************************************

00282                         * SWI FUNCTION VECTOR TABLE
00283A F87D      0194    A SWIVTB FDB   ZINCH-SWIVTB INCHNP
00284A F87F      01B1    A        FDB   ZOTCH1-SWIVTB OUTCH
00285A F881      01CB    A        FDB   ZPDTA1-SWIVTB PDATA1
00286A F883      01C3    A        FDB   ZPDATA-SWIVTB PDATA
00287A F885      0175    A        FDB   ZOT2HS-SWIVTB OUT2HS
00288A F887      0173    A        FDB   ZOT4HS-SWIVTB OUT4HS
00289A F889      01C0    A        FDB   ZPCRLF-SWIVTB PCRLF
00290A F88B      0179    A        FDB   ZSPACE-SWIVTB SPACE
00291A F88D      0055    A        FDB   ZMONTR-SWIVTB MONITR
00292A F88F      017D    A        FDB   ZVSWTH-SWIVTB VCTRSW
00293A F891      0256    A        FDB   ZBKPNT-SWIVTB BREAKPOINT
00294A F893      01D1    A        FDB   ZPAUSE-SWIVTB TASK PAUSE

00296A F895 6A   8D E6F7   SWIR   DEC   SWICNT,PCR UP "SWI" LEVEL FOR TRACE
00297A F899 17   0225 FAC1        LBSR  LDDP       SETUP PAGE AND VERIFY STACK
00298                       * CHECK FOR BREAKPOINT TRAP
00299A F89C EE   6A       A       LDU   10,S       LOAD PROGRAM COUNTER
00300A F89E 33   5F       A       LEAU  -1,U       BACK TO SWI ADDRESS
00301A F8A0 0D   FB       A       TST   SWIBFL     ? THIS "SWI" BREAKPOINT
00302A F8A2 26   11   F8B5        BNE   SWIDNE     BRANCH IF SO TO LET THROUGH
00303A F8A4 17   069B FF42        LBSR  CBKLDR     OBTAIN BREAKPOINT POINTERS
00304A F8A7 50                    NEGB             OBTAIN POSITIVE COUNT
00305A F8A8 5A             SWILP  DECB             COUNT DOWN
00306A F8A9 2B   0A   F8B5        BMI   SWIDNE     BRANCH WHEN DONE
00307A F8AB 11A3 A1       A       CMPU  ,Y++       ? WAS THIS A BREAKPOINT
00308A F8AE 26   F8   F8A8        BNE   SWILP      BRANCH IF NOT
00309A F8B0 EF   6A       A       STU   10,S       SET PROGRAM COUNTER BACK
00310A F8B2 16   021E EAD3        LBRA  ZBKPNT     GO DO BREAKPOINT
00311A F8B5 0F   FB       A SWIDNE CLR  SWIBFL     CLEAR IN CASE SET
00312A F8B7 37   06       A       PULU  D          OBTAIN FUNCTION BYTE, UP PC
00313A F8B9 C1   0B       A       CMPB  #NUMFUN    ? TOO HIGH
00314A F8BB 1022 020F FACE        LBHI  ERROR      YES, DO BREAKPOINT
00315A F8BF EF   6A       A       STU   10,S       BUMP PROGRAM COUNTER PAST SWI
00316A F8C1 58                    ASLB             FUNCTION CODE TIMES TWO
00317A F8C2 33   8C B8            LEAU  SWIVTB,PCR OBTAIN VECTOR BRANCH ADDRESS
00318A F8C5 EC   C5       A       LDD   B,U        LOAD OFFSET
00319A F8C7 6E   CB       A       JMP   D,U        JUMP TO ROUTINE

00321                       ***********************************************
00322                       * REGISTERS TO FUNCTION ROUTINES:
00323                       *  DP-> WORK AREA PAGE
00324                       *  D,Y,U=UNRELIABLE          X=AS CALLED FROM USER
```

```
00325                          *  S=AS FROM SWI INTERRUPT
00326                          **************************************************

00328                          ***************************************************
00329                          *              [SWI FUNCTION 8]
00330                          *                MONITOR ENTRY
00331                          *  FIREUP THE ASSIST09 MONITOR.
00332                          *  THE STACK WITH ITS VALUES FOR THE DIRECT PAGE
00333                          *  REGISTER AND CONDITION CODE FLAGS ARE USED AS IS.
00334                          *   1) INITIALIZE CONSOLE I/O
00335                          *   2) OPTIONALLY PRINT SIGNON
00336                          *   3) INITIALIZE PTM FOR SINGLE STEPPING
00337                          *   4) ENTER COMMAND PROCESSOR
00338                          * INPUT: A=0 INIT CONSOLE AND PRINT STARTUP MESSAGE
00339                          *        A#0 OMIT CONSOLE INIT AND STARTUP MESSAGE
00340                          ***************************************************

00342A F8C9      41       A SIGNON FCC    /ASSIST09/SIGNON EYE-CATCHER
00343A F8D1      04       A        FCB    EOT

00345A F8D2 10DF 97       A ZMONTR STS    RSTACK     SAVE FOR BAD STACK RECOVERY
00346A F8D5 6D   61       A        TST    1,S        ? INIT CONSOLE AND SEND MSG
00347A F8D7 26   0D   F8E6         BNE    ZMONT2     BRANCH IF NOT
00348A F8D9 AD   9D E6F9           JSR    [VECTAB+.CION,PCR] READY CONSOLE INPUT
00349A F8DD AD   9D E6FB           JSR    [VECTAB+.COON,PCR] READY CONSOLE OUTPUT
00350A F8E1 30   8C E5             LEAX   SIGNON,PCR READY SIGNON EYE-CATCHER
00351A F8E4 3F                     SWI               PERFORM
00352A F8E5      03       A        FCB    PDATA      PRINT STRING
00353A F8E6 9E   F6       A ZMONT2 LDX    VECTAB+.PTM LOAD PTM ADDRESS
00354A F8E8 27   0D   F8F7         BEQ    CMD        BRANCH IF NOT TO USE A PTM
00355A F8EA 6F   02       A        CLR    PTMTM1-PTM,X SET LATCH TO CLEAR RESET
00356A F8EC 6F   03       A        CLR    PTMTM1+1-PTM,X AND SET GATE HIGH
00357A F8EE CC   01A6     A        LDD    #$01A6     SETUP TIMER 1 MODE
00358A F8F1 A7   01       A        STA    PTMC2-PTM,X SETUP FOR CONTROL REGISTER1
00359A F8F3 E7   84       A        STB    PTMC13-PTM,X SET OUTPUT ENABLED/
00360                          *    SINGLE SHOT/ DUAL 8 BIT/INTERNAL MODE/OPERATE
00361A F8F5 6F   01       A        CLR    PTMC2-PTM,X SET CR2 BACK TO RESET FORM
00362                          * FALL INTO COMMAND PROCESSOR

00364                          ***************************************************
00365                          *           COMMAND HANDLER
00366                          *  BREAKPOINTS ARE REMOVED AT THIS TIME.
00367                          *  PROMPT FOR A COMMAND, AND STORE ALL CHARACTERS
00368                          *  UNTIL A SEPARATOR ON THE STACK.
00369                          *  SEARCH FOR FIRST MATCHING COMMAND SUBSET,
00370                          *  CALL IT OR GIVE '?' RESPONSE.
00371                          *  DURING COMMAND SEARCH:
00372                          *      B=OFFSET TO NEXT ENTRY ON X
00373                          *      U=SAVED S
00374                          *      U-1=ENTRY SIZE+2
00375                          *      U-2=VALID NUMBER FLAG (>=0 VALID)/COMPARE CNT
00376                          *      U-3=CARRIAGE RETURN FLAG (0=CR HAS BEEN DONE)
00377                          *      U-4=START OF COMMAND STORE
00378                          *      S+0=END OF COMMAND STORE
```

```
00379                            ***********************************************
00380A F8F7 3F                   CMD    SWI                TO NEW LINE
00381A F8F8       05       A            FCB     PCRLF      FUNCTION
00382                            * DISARM THE BREAKPOINTS
00383A F8F9 17    0646 FF42 CMDNEP LBSR  CBKLDR     OBTAIN BREAKPOINT POINTERS
00384A F8FC 2A    0C   F90A             BPL     CMDNOL     BRANCH IF NOT ARMED OR NONE
00385A F8FE 50                          NEGB               MAKE POSITIVE
00386A F8FF D7    FA       A            STB     BKPTCT     FLAG AS DISARMED
00387A F901 5A                   CMDDDL DECB               ? FINISHED
00388A F902 2B    06   F90A             BMI     CMDNOL     BRANCH IF SO
00389A F904 A6    30       A            LDA     -NUMBKP*2,Y LOAD OPCODE STORED
00390A F906 A7    B1       A            STA     [,Y++]     STORE BACK OVER "SWI"
00391A F908 20    F7   F901             BRA     CMDDDL     LOOP UNTIL DONE
00392A F90A AE    6A       A CMDNOL LDX     10,S       LOAD USERS PROGRAM COUNTER
00393A F90C 9F    93       A            STX     PCNTER     SAVE FOR EXPRESSION ANALYZER
00394A F90E 86    3E       A            LDA     #PROMPT    LOAD PROMPT CHARACTER
00395A F910 3F                          SWI                SEND TO OUTPUT HANDLER
00396A F911       01       A            FCB     OUTCH      FUNCTION
00397A F912 33    E4       A            LEAU    ,S         REMEMBER STACK RESTORE ADDRESS
00398A F914 DF    95       A            STU     PSTACK     REMEMBER STACK FOR ERROR USE
00399A F916 4F                          CLRA               PREPARE ZERO
00400A F917 5F                          CLRB               PREPARE ZERO
00401A F918 DD    9B       A            STD     NUMBER     CLEAR NUMBER BUILD AREA
00402A F91A DD    8F       A            STD     MISFLG     CLEAR MISCEL. AND SWICNT FLAGS
00403A F91C DD    91       A            STD     TRACEC     CLEAR TRACE COUNT
00404A F91E C6    02       A            LDAB    #2         SET D TO TWO
00405A F920 34    07       A            PSHS    D,CC       PLACE DEFAULTS ONTO STACK
00406                            * CHECK FOR "QUICK" COMMANDS.
00407A F922 17    0454 FD79             LBSR    READ       OBTAIN FIRST CHARACTER
00408A F925 30    8D 0581               LEAX    CDOT+2,PCR PRESET FOR SINGLE TRACE
00409A F929 81    2E       A            CMPA    #'.        ? QUICK TRACE
00410A F92B 27    5A   F987             BEQ     CMDXQT     BRANCH EQUAL FOR TRACE ONE
00411A F92D 30    8D 04E9               LEAX    CMPADP+2,PCR READY MEMORY ENTRY POINT
00412A F931 81    2F       A            CMPA    #'/        ? OPEN LAST USED MEMORY
00413A F933 27    52   F987             BEQ     CMDXQT     BRANCH TO DO IT IF SO
00414                            * PROCESS NEXT CHARACTER
00415A F935 81    20       A CMD2   CMPA    #'         ? BLANK OR DELIMITER
00416A F937 23    14   F94D             BLS     CMDGOT     BRANCH YES, WE HAVE IT
00417A F939 34    02       A            PSHS    A          BUILD ONTO STACK
00418A F93B 6C    5F       A            INC     -1,U       COUNT THIS CHARACTER
00419A F93D 81    2F       A            CMPA    #'/        ? MEMORY COMMAND
00420A F93F 27    4F   F990             BEQ     CMDMEM     BRANCH IF SO
00421A F941 17    040B FD4F             LBSR    BLDHXC     TREAT AS HEX VALUE
00422A F944 27    02   F948             BEQ     CMD3       BRANCH IF STILL VALID NUMBER
00423A F946 6A    5E       A            DEC     -2,U       FLAG AS INVALID NUMBER
00424A F948 17    042E FD79 CMD3   LBSR    READ       OBTAIN NEXT CHARACTER
00425A F94B 20    E8   F935             BRA     CMD2       TEST NEXT CHARACTER
00426                            * GOT COMMAND, NOW SEARCH TABLES
00427A F94D 80    0D       A CMDGOT SUBA    #CR        SET ZERO IF CARRIAGE RETURN
00428A F94F A7    5D       A            STA     -3,U       SETUP FLAG
00429A F951 9E    C4       A            LDX     VECTAB+.CMDL1 START WITH FIRST CMD LIST
00430A F953 E6    80       A CMDSCH LDB     ,X+        LOAD ENTRY LENGTH
00431A F955 2A    10   F967             BPL     CMDSME     BRANCH IF NOT LIST END
00432A F957 9E    EE       A            LDX     VECTAB+.CMDL2 NOW TO SECOND CMD LIST
00433A F959 5C                          INCB               ? TO CONTINUE TO DEFAULT LIST
00434A F95A 27    F7   F953             BEQ     CMDSCH     BRANCH IF SO
00435A F95C 10DE  95       A CMDBAD LDS     PSTACK     RESTORE STACK
00436A F95F 30    8D 015A               LEAX    ERRMSG,PCR POINT TO ERROR STRING
```

```
00437A F963 3F                 SWI              SEND OUT
00438A F964    02      A       FCB     PDATA1   TO CONSOLE
00439A F965 20 90   F8F7       BRA     CMD      AND TRY AGAIN
00440                     * SEARCH NEXT ENTRY
00441A F967 5A            CMDSME DECB           TAKE ACCOUNT OF LENGTH BYTE
00442A F968 E1 5F      A       CMPB    -1,U     ? ENTERED LONGER THAN ENTRY
00443A F96A 24 03   F96F       BHS     CMDSIZ   BRANCH IF NOT TOO LONG
00444A F96C 3A            CMDFLS ABX            SKIP TO NEXT ENTRY
00445A F96D 20 E4   F953       BRA     CMDSCH   AND TRY NEXT
00446A F96F 31 5D      A  CMDSIZ LEAY  -3,U     PREPARE TO COMPARE
00447A F971 A6 5F      A       LDA     -1,U     LOAD SIZE+2
00448A F973 80 02      A       SUBA    #2       TO ACTUAL SIZE ENTERED
00449A F975 A7 5E      A       STA     -2,U     SAVE SIZE FOR COUNTDOWN
00450A F977 5A            CMDCMP DECB           DOWN ONE BYTE
00451A F978 A6 80      A       LDA     ,X+      NEXT COMMAND CHARACTER
00452A F97A A1 A2      A       CMPA    ,-Y      ? SAME AS THAT ENTERED
00453A F97C 26 EE   F96C       BNE     CMDFLS   BRANCH TO FLUSH  IF NOT
00454A F97E 6A 5E      A       DEC     -2,U     COUNT DOWN LENGTH OF ENTRY
00455A F980 26 F5   F977       BNE     CMDCMP   BRANCH IF MORE TO TEST
00456A F982 3A                 ABX              TO NEXT ENTRY
00457A F983 EC 1E      A       LDD     -2,X     LOAD OFFSET
00458A F985 30 8B      A       LEAX    D,X      COMPUTE ROUTINE ADDRESS+2
00459A F987 6D 5D      A  CMDXQT TST   -3,U     SET CC FOR CARRIAGE RETURN TEST
00460A F989 32 C4      A       LEAS    ,U       DELETE STACK WORK AREA
00461A F98B AD 1E      A       JSR     -2,X     CALL COMMAND
00462A F98D 16 FF7A F90A       LBRA    CMDNOL   GO GET NEXT COMMAND
00463A F990 6D 5E      A  CMDMEM TST   -2,U     ? VALID HEX NUMBER ENTERED
00464A F992 2B C8   F95C       BMI     CMDBAD   BRANCH ERROR IF NOT
00465A F994 30 88 AE   A       LEAX    <CMEMN-CMPADP,X TO DIFFERENT ENTRY
00466A F997 DC 9B      A       LDD     NUMBER   LOAD NUMBER ENTERED
00467A F999 20 EC   F987       BRA     CMDXQT   AND ENTER MEMORY COMMAND

00469                     ** COMMANDS ARE ENTERED AS A SUBROUTINE WITH:
00470                     **     DPR->ASSIST09 DIRECT PAGE WORK AREA
00471                     **     Z=1 CARRIAGE RETURN ENTERED
00472                     **     Z=0 NON CARRIAGE RETURN DELIMITER
00473                     **     S=NORMAL RETURN ADDRESS
00474                     ** THE LABEL "CMDBAD" MAY BE ENTERED TO ISSUE AN
00475                     ** AN ERROR FLAG (*).


00477                     ***********************************************************
00478                     *        ASSIST09 COMMAND TABLES
00479                     *  THESE ARE THE DEFAULT COMMAND TABLES.  EXTERNAL
00480                     *  TABLES OF THE SAME FORMAT MAY EXTEND/REPLACE
00481                     *  THESE BY USING THE VECTOR SWAP FUNCTION.
00482                     *
00483                     * ENTRY FORMAT:
00484                     *      +0...TOTAL SIZE OF ENTRY (INCLUDING THIS BYTE)
00485                     *      +1...COMMAND STRING
00486                     *      +N...TWO BYTE OFFSET TO COMMAND (ENTRYADDR-*)
00487                     *
00488                     *  THE TABLES TERMINATE WITH A ONE BYTE -1 OR -2.
00489                     *  THE -1 CONTINUES THE COMMAND SEARCH WITH THE
00490                     *         SECOND COMMAND TABLE.
00491                     *  THE -2 TERMINATES COMMAND SEARCHES.
00492                     ***********************************************************
```

```
00494                            * THIS IS THE DEFAULT LIST FOR THE SECOND COMMAND
00495                            * LIST ENTRY.
00496A F99B     FE         A CMDTB2 FCB    -2        STOP COMMAND SEARCHES

00498                            * THIS IS THE DEFAULT LIST FOR THE FIRST COMMAND
00499                            * LIST ENTRY.
00500           F99C       A CMDTBL EQU    *         MONITOR COMMAND TABLE
00501A F99C     04         A        FCB    4
00502A F99D     42         A        FCC    /B/       'BREAKPOINT' COMMAND
00503A F99E     054D       A        FDB    CBKPT-*
00504A F9A0     04         A        FCB    4
00505A F9A1     43         A        FCC    /C/       'CALL' COMMAND
00506A F9A2     0417       A        FDB    CCALL-*
00507A F9A4     04         A        FCB    4
00508A F9A5     44         A        FCC    /D/       'DISPLAY' COMMAND
00509A F9A6     049D       A        FDB    CDISP-*
00510A F9A8     04         A        FCB    4
00511A F9A9     45         A        FCC    /E/       'ENCODE' COMMAND
00512A F9AA     059F       A        FDB    CENCDE-*
00513A F9AC     04         A        FCB    4
00514A F9AD     47         A        FCC    /G/       'GO' COMMAND
00515A F9AE     03D2       A        FDB    CGO-*
00516A F9B0     04         A        FCB    4
00517A F9B1     4C         A        FCC    /L/       'LOAD' COMMAND
00518A F9B2     04DD       A        FDB    CLOAD-*
00519A F9B4     04         A        FCB    4
00520A F9B5     4D         A        FCC    /M/       'MEMORY' COMMAND
00521A F9B6     040D       A        FDB    CMEM-*
00522A F9B8     04         A        FCB    4
00523A F9B9     4E         A        FCC    /N/       'NULLS' COMMAND
00524A F9BA     04FD       A        FDB    CNULLS-*
00525A F9BC     04         A        FCB    4
00526A F9BD     4F         A        FCC    /O/       'OFFSET' COMMAND
00527A F9BE     050A       A        FDB    COFFS-*
00528A F9C0     04         A        FCB    4
00529A F9C1     50         A        FCC    /P/       'PUNCH' COMMAND
00530A F9C2     04AF       A        FDB    CPUNCH-*
00531A F9C4     04         A        FCB    4
00532A F9C5     52         A        FCC    /R/       'REGISTERS' COMMAND
00533A F9C6     0284       A        FDB    CREG-*
00534A F9C8     04         A        FCB    4
00535A F9C9     53         A        FCC    /S/       'STLEVEL' COMMAND
00536A F9CA     04F2       A        FDB    CSTLEV-*
00537A F9CC     04         A        FCB    4
00538A F9CD     54         A        FCC    /T/       'TRACE' COMMAND
00539A F9CE     04D6       A        FDB    CTRACE-*
00540A F9D0     04         A        FCB    4
00541A F9D1     56         A        FCC    /V/       'VERIFY' COMMAND
00542A F9D2     04CF       A        FDB    CVER-*
00543A F9D4     04         A        FCB    4
00544A F9D5     57         A        FCC    /W/       'WINDOW' COMMAND
00545A F9D6     0468       A        FDB    CWINDO-*
00546A F9D8     FF         A        FCB    -1        END, CONTINUE WITH THE SECOND

00548                            *************************************************
00549                            *              [SWI FUNCTIONS 4 AND 5]
```

```
00550                          *       4 - OUT2HS - DECODE BYTE TO HEX AND ADD SPACE
00551                          *       5 - OUT4HS - DECODE WORD TO HEX AND ADD SPACE
00552                          * INPUT: X->BYTE OR WORD TO DECODE
00553                          * OUTPUT: CHARACTERS SENT TO OUTPUT HANDLER
00554                          *          X->NEXT BYTE OR WORD
00555                          ***************************************************

00557A F9D9 A6    80      A ZOUT2H LDA   ,X+        LOAD NEXT BYTE
00558A F9DB 34    06      A        PSHS  D          SAVE - DO NOT REREAD
00559A F9DD C6    10      A        LDB   #16        SHIFT BY 4 BITS
00560A F9DF 3D                     MUL              WITH MULTIPLY
00561A F9E0 8D    04   F9E6        BSR   ZOUTHX     SEND OUT AS HEX
00562A F9E2 35    06      A        PULS  D          RESTORE BYTES
00563A F9E4 84    0F      A        ANDA  #$0F       ISOLATE RIGHT HEX
00564A F9E6 8B    90      A ZOUTHX ADDA  #$90       PREPARE A-F ADJUST
00565A F9E8 19                     DAA              ADJUST
00566A F9E9 89    40      A        ADCA  #$40       PREPARE CHARACTER BITS
00567A F9EB 19                     DAA              ADJUST
00568A F9EC 6E    9D E5EE   SEND   JMP   [VECTAB+.CODTA,PCR] SEND TO OUT HANDLER

00570A F9F0 8D    E7   F9D9 ZOT4HS BSR   ZOUT2H     CONVERT FIRST BYTE
00571A F9F2 8D    E5   F9D9 ZOT2HS BSR   ZOUT2H     CONVERT BYTE TO HEX
00572A F9F4 AF    64      A        STX   4,S        UPDATE USERS X REGISTER
00573                          * FALL INTO SPACE ROUTINE


00575                          ***************************************************
00576                          *              [SWI FUNCTION 7]
00577                          *        SPACE - SEND BLANK TO OUTPUT HANDLER
00578                          * INPUT: NONE
00579                          * OUTPUT: BLANK SEND TO CONSOLE HANDLER
00580                          ***************************************************
00581A F9F6 86    20      A ZSPACE LDA   #'         LOAD BLANK
00582A F9F8 20    3D   FA37        BRA   ZOTCH2     SEND AND RETURN


00584                          ***************************************************
00585                          *              [SWI FUNCTION 9]
00586                          *           SWAP VECTOR TABLE ENTRY
00587                          * INPUT: A=VECTOR TABLE CODE (OFFSET)
00588                          *        X=0 OR REPLACEMENT VALUE
00589                          * OUTPUT: X=PREVIOUS VALUE
00590                          ***************************************************
00591A F9FA A6    61      A ZVSWTH LDA   1,S        LOAD REQUESTERS A
00592A F9FC 81    34      A        CMPA  #HIVTR     ? SUB-CODE TOO HIGH
00593A F9FE 22    39   FA39        BHI   ZOTCH3     IGNORE CALL IF SO
00594A FA00 109E  C2      A        LDY   VECTAB+.AVTBL LOAD VECTOR TABLE ADDRESS
00595A FA03 EE    A6      A        LDU   A,Y        U=OLD ENTRY
00596A FA05 EF    64      A        STU   4,S        RETURN OLD VALUE TO CALLERS X
00597A FA07 AF    7E      A        STX   -2,S       ? X=0
00598A FA09 27    2E   FA39        BEQ   ZOTCH3     YES, DO NOT CHANGE ENTRY
00599A FA0B AF    A6      A        STX   A,Y        REPLACE ENTRY
00600A FA0D 20    2A   FA39        BRA   ZOTCH3     RETURN FROM SWI
00601                          *D
```

```
00603                           ************************************************
00604                           *                 [SWI FUNCTION 0]
00605                           *   INCHNP - OBTAIN INPUT CHAR IN A (NO PARITY)
00606                           *  NULLS AND RUBOUTS ARE IGNORED.
00607                           *  AUTOMATIC LINE FEED IS SENT UPON RECIEVING A
00608                           *      CARRIAGE RETURN.
00609                           *  UNLESS WE ARE LOADING FROM TAPE.
00610                           ************************************************
00611A FA0F 8D    5D    FA6E ZINCHP BSR    XQPAUS    RELEASE PROCESSOR
00612A FA11 8D    5F    FA72 ZINCH  BSR    XQCIDT    CALL INPUT DATA APPENDAGE
00613A FA13 24    FA    FA0F        BCC    ZINCHP    LOOP IF NONE AVAILABLE
00614A FA15 4D                      TSTA             ? TEST FOR NULL
00615A FA16 27    F9    FA11        BEQ    ZINCH     IGNORE NULL
00616A FA18 81    7F       A        CMPA   #$7F      ? RUBOUT
00617A FA1A 27    F5    FA11        BEQ    ZINCH     BRANCH YES TO IGNORE
00618A FA1C A7    61       A        STA    1,S       STORE INTO CALLERS A
00619A FA1E 0D    8F       A        TST    MISFLG    ? LOAD IN PROGRESS
00620A FA20 26    17    FA39        BNE    ZOTCH3    BRANCH IF SO TO NOT ECHO
00621A FA22 81    0D       A        CMPA   #CR       ? CARRIAGE RETURN
00622A FA24 26    04    FA2A        BNE    ZIN2      NO, TEST ECHO BYTE
00623A FA26 86    0A       A        LDA    #LF       LOAD LINE FEED
00624A FA28 8D    C2    F9EC        BSR    SEND      ALWAYS ECHO LINE FEED
00625A FA2A 0D    F4       A ZIN2   TST    VECTAB+.ECHO ? ECHO DESIRED
00626A FA2C 26    0B    FA39        BNE    ZOTCH3    NO, RETURN
00627                           * FALL THROUGH TO OUTCH


00629                           ************************************************
00630                           *              [SWI FUNCTION 1]
00631                           *          OUTCH - OUTPUT CHARACTER FROM A
00632                           *  INPUT:  NONE
00633                           *  OUTPUT: IF LINEFEED IS THE OUTPUT CHARACTER THEN
00634                           *            C=0 NO CTL-X RECIEVED, C=1 CTL-X RECIEVED
00635                           ************************************************
00636A FA2E A6    61       A ZOTCH1 LDA    1,S       LOAD CHARACTER TO SEND
00637A FA30 30    8C 09             LEAX   <ZPCRLS,PCR DEFAULT FOR LINE FEED
00638A FA33 81    0A       A        CMPA   #LF       ? LINE FEED
00639A FA35 27    0F    FA46        BEQ    ZPDTLP    BRANCH TO CHECK PAUSE IF SO
00640A FA37 8D    B3    F9EC ZOTCH2 BSR    SEND      SEND TO OUTPUT ROUTINE
00641A FA39 0C    90       A ZOTCH3 INC    SWICNT    BUMP UP "SWI" TRACE NEST LEVEL
00642A FA3B 3B                      RTI              RETURN FROM "SWI" FUNCTION


00644                           ************************************************
00645                           *              [SWI FUNCTION 6]
00646                           *       PCRLF - SEND CR/LF TO CONSOLE HANDLER
00647                           *  INPUT: NONE
00648                           *  OUTPUT: CR AND LF SENT TO HANDLER
00649                           *            C=0 NO CTL-X, C=1 CTL-X RECIEVED
00650                           ************************************************

00652A FA3C       04       A ZPCRLS FCB    EOT       NULL STRING

00654A FA3D 30    8C FC      ZPCRLF LEAX   ZPCRLS,PCR READY CR,LF STRING
00655                           * FALL INTO CR/LF CODE
```

```
00657                          ***********************************************
00658                          *               [SWI FUNCTION 3]
00659                          *          PDATA - OUTPUT CR/LF AND STRING
00660                          * INPUT: X->STRING
00661                          * OUTPUT: CR/LF AND STRING SENT TO OUTPUT CONSOLE
00662                          *            HANDLER.
00663                          *        C=0 NO CTL-X, C=1 CTL-X RECIEVED
00664                          * NOTE: LINE FEED MUST FOLLOW CARRIAGE RETURN FOR
00665                          *           PROPER PUNCH DATA.
00666                          ***********************************************
00667A FA40 86    0D       A ZPDATA LDA   #CR       LOAD CARRIAGE RETURN
00668A FA42 8D    A8    F9EC        BSR   SEND      SEND IT
00669A FA44 86    0A       A        LDA   #LF       LOAD LINE FEED
00670                          * FALL INTO PDATA1


00672                          ***********************************************
00673                          *               [SWI FUNCTION 2]
00674                          *          PDATA1 - OUTPUT STRING TILL EOT ($04)
00675                          *  THIS ROUTINE PAUSES IF AN INPUT BYTE BECOMES
00676                          *  AVAILABLE DURING OUTPUT TRANSMISSION UNTIL A
00677                          *  SECOND IS RECIEVED.
00678                          * INPUT: X->STRING
00679                          * OUTPUT: STRING SENT TO OUTPUT CONSOLE DRIVER
00680                          *          C=0 NO CTL-X, C=1 CTL-X RECIEVED
00681                          ***********************************************
00682A FA46 8D    A4    F9EC ZPDTLP BSR   SEND      SEND CHARACTER TO DRIVER
00683A FA48 A6    80       A ZPDTA1 LDA   ,X+       LOAD NEXT CHARACTER
00684A FA4A 81    04       A        CMPA  #EOT      ? EOT
00685A FA4C 26    F8    FA46        BNE   ZPDTLP    LOOP IF NOT
00686                          * FALL INTO PAUSE CHECK FUNCTION


00688                          ********************************************
00689                          *               [SWI FUNCTION 12]
00690                          *       PAUSE - RETURN TO TASK DISPATCHING AND CHECK
00691                          *               FOR FREEZE CONDITION OR CTL-X BREAK
00692                          *  THIS FUNCTION ENTERS THE TASK PAUSE HANDLER SO
00693                          *  OPTIONALLY OTHER 6809 PROCESSES MAY GAIN CONTROL.
00694                          *  UPON RETURN, CHECK FOR A 'FREEZE' CONDITION
00695                          *  WITH A RESULTING WAIT LOOP, OR CONDITION CODE
00696                          *  RETURN IF A CONTROL-X IS ENTERED FROM THE INPUT
00697                          *  HANDLER.
00698                          * OUTPUT: C=1 IF CTL-X HAS ENTERED, C=0 OTHERWISE
00699                          ********************************************
00700A FA4E 8D    1E    FA6E ZPAUSE BSR   XQPAUS    RELEASE CONTROL AT EVERY LINE
00701A FA50 8D    06    FA58        BSR   CHKABT    CHECK FOR FREEZE OR ABORT
00702A FA52 1F    A9       A        TFR   CC,B      PREPARE TO REPLACE CC
00703A FA54 E7    E4       A        STB   ,S        OVERLAY OLD ONE ON STACK
00704A FA56 20    E1    FA39        BRA   ZOTCH3    RETURN FROM "SWI"

00706                          * CHKABT - SCAN FOR INPUT PAUSE/ABORT DURING OUTPUT
00707                          * OUTPUT: C=0 OK, C=1 ABORT (CTL-X ISSUED)
00708                          * VOLATILE: U,X,D
00709A FA58 8D    18    FA72 CHKABT BSR   XQCIDT    ATTEMPT INPUT
00710A FA5A 24    05    FA61        BCC   CHKRTN    BRANCH NO TO RETURN
```

```
00711A FA5C 81     18       A         CMPA   #CAN       ? CTL-X FOR ABORT
00712A FA5E 26     02    FA62         BNE    CHKWT      BRANCH NO TO PAUSE
00713A FA60 53                CHKSEC  COMB              SET CARRY
00714A FA61 39                CHKRTN  RTS               RETURN TO CALLER WITH CC SET
00715A FA62 8D     0A    FA6E CHKWT   BSR    XQPAUS     PAUSE FOR A MOMENT
00716A FA64 8D     0C    FA72         BSR    XQCIDT     ? KEY FOR START
00717A FA66 24     FA    FA62         BCC    CHKWT      LOOP UNTIL RECIEVED
00718A FA68 81     18       A         CMPA   #CAN       ? ABORT SIGNALED FROM WAIT
00719A FA6A 27     F4    FA60         BEQ    CHKSEC     BRANCH YES
00720A FA6C 4F                        CLRA              SET C=0 FOR NO ABORT
00721A FA6D 39                        RTS               AND RETURN

00723                         * SAVE MEMORY WITH JUMPS
00724A FA6E 6E     9D E578    XQPAUS  JMP    [VECTAB+.PAUSE,PCR] TO PAUSE ROUTINE
00725A FA72 AD     9D E562    XQCIDT  JSR    [VECTAB+.CIDTA,PCR] TO INPUT ROUTINE
00726A FA76 84     7F       A         ANDA   #$7F       STRIP PARITY
00727A FA78 39                        RTS               RETURN TO CALLER

00729                         ***********************************************
00730                         *         NMI DEFAULT INTERRUPT HANDLER
00731                         *  THE NMI HANDLER IS USED FOR TRACING INSTRUCTIONS.
00732                         *  TRACE PRINTOUTS OCCUR ONLY AS LONG AS THE STACK
00733                         *  TRACE LEVEL IS NOT BREACHED BY FALLING BELOW IT.
00734                         *  TRACING CONTINUES UNTIL THE COUNT TURNS ZERO OR
00735                         *  A CTL-X IS ENTERED FROM THE INPUT CONSOLE DEVICE.
00736                         ***********************************************

00738A FA79        4F       A MSHOWP  FCB    'O,'P,'-,EOT OPCODE PREP

00740A FA7D 8D     42    FAC1 NMIR    BSR    LDDP       LOAD PAGE AND VERIFY STACK
00741A FA7F 0D     8F       A         TST    MISFLG     ? THRU A BREAKPOINT
00742A FA81 26     34    FAB7         BNE    NMICON     BRANCH IF SO TO CONTINUE
00743A FA83 0D     90       A         TST    SWICNT     ? INHIBIT "SWI" DURING TRACE
00744A FA85 2B     29    FAB0         BMI    NMITRC     BRANCH YES
00745A FA87 30     6C       A         LEAX   12,S       OBTAIN USERS STACK POINTER
00746A FA89 9C     F8       A         CMPX   SLEVEL     ? TO TRACE HERE
00747A FA8B 25     23    FAB0         BLO    NMITRC     BRANCH IF TOO LOW TO DISPLAY
00748A FA8D 30     8C E9              LEAX   MSHOWP,PCR LOAD OP PREP
00749A FA90 3F                        SWI               SEND TO CONSOLE
00750A FA91        02       A         FCB    PDATA1     FUNCTION
00751A FA92 09     8E       A         ROL    DELIM      SAVE CARRY BIT
00752A FA94 30     8D E501            LEAX   LASTOP,PCR POINT TO LAST OP
00753A FA98 3F                        SWI               SEND OUT AS HEX
00754A FA99        05       A         FCB    OUT4HS     FUNCTION
00755A FA9A 8D     17    FAB3         BSR    REGPRS     FOLLOW MEMORY WITH REGISTERS
00756A FA9C 25     37    FAD5         BCS    ZBKCMD     BRANCH IF "CANCEL"
00757A FA9E 06     8E       A         ROR    DELIM      RESTORE CARRY BIT
00758A FAA0 25     33    FAD5         BCS    ZBKCMD     BRANCH IF "CANCEL"
00759A FAA2 9E     91       A         LDX    TRACEC     LOAD TRACE COUNT
00760A FAA4 27     2F    FAD5         BEQ    ZBKCMD     IF ZERO TO COMMAND HANDLER
00761A FAA6 30     1F       A         LEAX   -1,X       MINUS ONE
00762A FAA8 9F     91       A         STX    TRACEC     REFRESH
00763A FAAA 27     29    FAD5         BEQ    ZBKCMD     STOP TRACE WHEN ZERO
00764A FAAC 8D     AA    FA58         BSR    CHKABT     ? ABORT THE TRACE
00765A FAAE 25     25    FAD5         BCS    ZBKCMD     BRANCH YES TO COMMAND HANDLER
```

```
00766A FAB0 16   03F7 FEAA NMITRC LBRA   CTRCE3   NO, TRACE ANOTHER INSTRUCTION

00768A FAB3 17   01B9 FC6F REGPRS LBSR   REGPRT   PRINT REGISTERS AS FROM COMMAND
00769A FAB6 39                    RTS             RETURN TO CALLER

00771                      * JUST EXECUTED THRU A BRKPNT.  NOW CONTINUE NORMALLY
00772A FAB7 0F   8F      A NMICON CLR    MISFLG   CLEAR THRU FLAG
00773A FAB9 17   02EB FDA7        LBSR   ARMBK2   ARM BREAKPOINTS
00774A FABC 3B             RTI    RTI             AND CONTINUE USERS PROGRAM

00776                      * LDDP - SETUP DIRECT PAGE REGISTER, VERIFY STACK.
00777                      * AN INVALID STACK CAUSES A RETURN TO THE COMMAND
00778                      * HANDLER.
00779                      * INPUT: FULLY STACKED REGISTERS FROM AN INTERRUPT
00780                      * OUTPUT: DPR LOADED TO WORK PAGE

00782A FABD      3F      A ERRMSG FCB    '?,BELL,$20,EOT ERROR RESPONSE

00784A FAC1 E6   8D E4D8   LDDP   LDB    BASEPG,PCR LOAD DIRECT PAGE HIGH BYTE
00785A FAC5 1F   9B      A        TFR    B,DP     SETUP DIRECT PAGE REGISTER
00786A FAC7 A1   63      A        CMPA   3,S      ? IS STACK VALID
00787A FAC9 27   25   FAF0        BEQ    RTS      YES, RETURN
00788A FACB 10DE 97      A        LDS    RSTACK   RESET TO INITIAL STACK POINTER
00789A FACE 30   8C EC     ERROR  LEAX   ERRMSG,PCR LOAD ERROR REPORT
00790A FAD1 3F                    SWI             SEND OUT BEFORE REGISTERS
00791A FAD2      03      A        FCB    PDATA    ON NEXT LINE
00792                      * FALL INTO BREAKPOINT HANDLER


00794                      ***********************************************
00795                      *          [SWI FUNCTION 10]
00796                      *       BREAKPOINT PROGRAM FUNCTION
00797                      *  PRINT REGISTERS AND GO TO COMMAND HANLER
00798                      ***********************************************
00799A FAD3 8D   0E   FAB3 ZBKPNT BSR    REGPRS   PRINT OUT REGISTERS
00800A FAD5 16   FE21 F8F9 ZBKCMD LBRA   CMDNEP   NOW ENTER COMMAND HANDLER


00802                      ***********************************************
00803                      *   IRQ, RESERVED, SWI2 AND SWI3 INTERRUPT HANDLERS
00804                      *  THE DEFAULT HANDLING IS TO CAUSE A BREAKPOINT.
00805                      ***********************************************
00806            FAD8    A SWI2R  EQU    *        SWI2 ENTRY
00807            FAD8    A SWI3R  EQU    *        SWI3 ENTRY
00808            FAD8    A IRQR   EQU    *        IRQ ENTRY
00809A FAD8 8D   E7   FAC1 RSRVDR BSR    LDDP     SET BASE PAGE, VALIDATE STACK
00810A FADA 20   F7   FAD3        BRA    ZBKPNT   FORCE A BREAKPOINT


00812                      ******************************************
00813                      *        FIRQ HANDLER
00814                      *  JUST RETURN FOR THE FIRQ INTERRUPT
00815                      ******************************************
00816            FABC    A FIRQR  EQU    RTI      IMMEDIATE RETURN
```

```
00818                               *****************************************************
00819                               *       DEFAULT I/O DRIVERS
00820                               *****************************************************

00822                               * CIDTA - RETURN CONSOLE INPUT CHARACTER
00823                               * OUTPUT: C=0 IF NO DATA READY, C=1 A=CHARACTER
00824                               * U VOLATILE
00825A FADC DE    F0        A CIDTA  LDU    VECTAB+.ACIA LOAD ACIA ADDRESS
00826A FADE A6    C4        A        LDA    ,U         LOAD STATUS REGISTER
00827A FAE0 44                       LSRA              TEST RECIEVER REGISTER FLAG
00828A FAE1 24    02   FAE5          BCC    CIRTN      RETURN IF NOTHING
00829A FAE3 A6    41        A        LDA    1,U        LOAD DATA BYTE
00830A FAE5 39                CIRTN  RTS               RETURN TO CALLER

00832                               * CION - INPUT CONSOLE INITIALIZATION
00833                               * COON - OUTPUT CONSOLE INITIALIZATION
00834                               * A,X  VOLATILE
00835             FAE6      A CION   EQU    *
00836A FAE6 86    03        A COON   LDA    #3         RESET ACIA CODE
00837A FAE8 9E    F0        A        LDX    VECTAB+.ACIA LOAD ACIA ADDRESS
00838A FAEA A7    84        A        STA    ,X         STORE INTO STATUS REGISTER
00839A FAEC 86    51        A        LDA    #$51       SET CONTROL
00840A FAEE A7    84        A        STA    ,X         REGISTER UP
00841A FAF0 39                RTS    RTS               RETURN TO CALLER

00843                               * THE FOLLOWING HAVE NO DUTIES TO PERFORM
00844             FAF0      A CIOFF  EQU    RTS        CONSOLE INPUT OFF
00845             FAF0      A COOFF  EQU    RTS        CONSOLE OUTPUT OFF

00847                               * CODTA - OUTPUT CHARACTER TO CONSOLE DEVICE
00848                               * INPUT: A=CHARACTER TO SEND
00849                               * OUTPUT: CHAR SENT TO TERMINAL WITH PROPER PADDING
00850                               * ALL REGISTERS TRANSPARENT

00852A FAF1 34    47        A CODTA  PSHS   U,D,CC     SAVE REGISTERS,WORK BYTE
00853A FAF3 DE    F0        A        LDU    VECTAB+.ACIA ADDRESS ACIA
00854A FAF5 8D    1B   FB12          BSR    CODTAO     CALL OUTPUT CHAR SUBROTINE
00855A FAF7 81    10        A        CMPA   #DLE       ? DATA LINE ESCAPE
00856A FAF9 27    12   FB0D          BEQ    CODTRT     YES, RETURN
00857A FAFB D6    F2        A        LDB    VECTAB+.PAD DEFAULT TO CHAR PAD COUNT
00858A FAFD 81    0D        A        CMPA   #CR        ? CR
00859A FAFF 26    02   FB03          BNE    CODTPD     BRANCH NO
00860A FB01 D6    F3        A        LDB    VECTAB+.PAD+1 LOAD NEW LINE PAD COUNT
00861A FB03 4F                CODTPD CLRA              CREATE NULL
00862A FB04 E7    E4        A        STB    ,S         SAVE COUNT
00863A FB06       8C        A        FCB    SKIP2      ENTER LOOP
00864A FB07 8D    09   FB12 CODTLP BSR    CODTAO     SEND NULL
00865A FB09 6A    E4        A        DEC    ,S         ? FINISHED
00866A FB0B 2A    FA   FB07          BPL    CODTLP     NO, CONTINUE WITH MORE
00867A FB0D 35    C7        A CODTRT PULS   PC,U,D,CC RESTORE REGISTERS AND RETURN

00869A FB0F 17    FF5C FA6E CODTAD LBSR   XQPAUS     TEMPORARY GIVE UP CONTROL
00870A FB12 E6    C4        A CODTAO LDB    ,U         LOAD ACIA CONTROL REGISTER
00871A FB14 C5    02        A        BITB   #$02       ? TX REGISTER CLEAR
```

```
00872A FB16 27    F7    FB0F          BEQ    CODTAD    RELEASE CONTROL IF NOT
00873A FB18 A7    41       A          STA    1,U       STORE INTO DATA REGISTER
00874A FB1A 39                        RTS              RETURN TO CALLER
00875                        *E


00877                        * BSON - TURN ON READ/VERIFY/PUNCH MECHANISM
00878                        * A IS VOLATILE

00880A FB1B 86    11       A BSON     LDA    #$11      SET READ CODE
00881A FB1D 6D    66       A          TST    6,S       ? READ OR VERIFY
00882A FB1F 26    01    FB22          BNE    BSON2     BRANCH YES
00883A FB21 4C                        INCA             SET TO WRITE
00884A FB22 3F               BSON2    SWI              PERFORM OUTPUT
00885A FB23       01       A          FCB    OUTCH     FUNCTION
00886A FB24 0C    8F       A          INC    MISFLG    SET LOAD IN PROGRESS FLAG
00887A FB26 39                        RTS              RETURN TO CALLER


00889                        * BSOFF - TURN OFF READ/VERIFY/PUNCH MECHANISM
00890                        * A,X VOLATILE
00891A FB27 86    14       A BSOFF    LDA    #$14      TO DC4 - STOP
00892A FB29 3F                        SWI              SEND OUT
00893A FB2A       01       A          FCB    OUTCH     FUNCTION
00894A FB2B 4A                        DECA             CHANGE TO DC3 (X-OFF)
00895A FB2C 3F                        SWI              SEND OUT
00896A FB2D       01       A          FCB    OUTCH     FUNCTION
00897A FB2E 0A    8F       A          DEC    MISFLG    CLEAR LOAD IN PROGRESS FLAG
00898A FB30 8E    61A8     A          LDX    #25000    DELAY 1 SECOND (2MHZ CLOCK)
00899A FB33 30    1F       A BSOFLP   LEAX   -1,X      COUNT DOWN
00900A FB35 26    FC    FB33          BNE    BSOFLP    LOOP TILL DONE
00901A FB37 39                        RTS              RETURN TO CALLER


00903                        * BSDTA - READ/VERIFY/PUNCH HANDLER
00904                        * INPUT: S+6=CODE BYTE, VERIFY(-1),PUNCH(0),LOAD(1)
00905                        *        S+4=START ADDRESS
00906                        *        S+2=STOP ADDRESS
00907                        *        S+0=RETURN ADDRESS
00908                        * OUTPUT: Z=1 NORMAL COMPLETION, Z=0 INVALID LOAD/VER
00909                        * REGISTERS ARE VOLATILE

00911A FB38 EE    62       A BSDTA    LDU    2,S       U=TO ADDRESS OR OFFSET
00912A FB3A 6D    66       A          TST    6,S       ? PUNCH
00913A FB3C 27    54    FB92          BEQ    BSDPUN    BRANCH YES
00914                        * DURING READ/VERIFY: S+2=MSB ADDRESS SAVE BYTE
00915                        *                     S+1=BYTE COUNTER
00916                        *                     S+0=CHECKSUM
00917                        *                     U HOLDS OFFSET
00918A FB3E 32    7D       A          LEAS   -3,S      ROOM FOR WORK/COUNTER/CHECKSUM
00919A FB40 3F               BSDLD1   SWI              GET NEXT CHARACTER
00920A FB41       00       A          FCB    INCHNP    FUNCTION
00921A FB42 81    53       A BSDLD2   CMPA   #'S       ? START OF S1/S9
00922A FB44 26    FA    FB40          BNE    BSDLD1    BRANCH NOT
00923A FB46 3F                        SWI              GET NEXT CHARACTE[illegible]
```

```
00924A FB47      00        A        FCB    INCHNP   FUNCTION
00925A FB48 81   39        A        CMPA   #'9      ? HAVE S9
00926A FB4A 27   22     FB6E        BEQ    BSDSRT   YES, RETURN GOOD CODE
00927A FB4C 81   31        A        CMPA   #'1      ? HAVE NEW RECORD
00928A FB4E 26   F2     FB42        BNE    BSDLD2   BRANCH IF NOT
00929A FB50 6F   E4        A        CLR    ,S       CLEAR CHECKSUM
00930A FB52 8D   21     FB75        BSR    BYTE     OBTAIN BYTE COUNT
00931A FB54 E7   61        A        STB    1,S      SAVE FOR DECREMENT
00932                       * READ ADDRESS
00933A FB56 8D   1D     FB75        BSR    BYTE     OBTAIN HIGH VALUE
00934A FB58 E7   62        A        STB    2,S      SAVE IT
00935A FB5A 8D   19     FB75        BSR    BYTE     OBTAIN LOW VALUE
00936A FB5C A6   62        A        LDA    2,S      MAKE D=VALUE
00937A FB5E 31   CB        A        LEAY   D,U      Y=ADDRESS+OFFSET
00938                       * STORE TEXT
00939A FB60 8D   13     FB75 BSDNXT BSR    BYTE     NEXT BYTE
00940A FB62 27   0C     FB70        BEQ    BSDEOL   BRANCH IF CHECKSUM
00941A FB64 6D   69        A        TST    9,S      ? VERIFY ONLY
00942A FB66 2B   02     FB6A        BMI    BSDCMP   YES, ONLY COMPARE
00943A FB68 E7   A4        A        STB    ,Y       STORE INTO MEMORY
00944A FB6A E1   A0        A BSDCMP CMPB   ,Y+      ? VALID RAM
00945A FB6C 27   F2     FB60        BEQ    BSDNXT   YES, CONTINUE READING
00946A FB6E 35   92        A BSDSRT PULS   PC,X,A   RETURN WITH Z SET PROPER

00948A FB70 4C               BSDEOL INCA            ? VALID CHECKSUM
00949A FB71 27   CD     FB40        BEQ    BSDLD1   BRANCH YES
00950A FB73 20   F9     FB6E        BRA    BSDSRT   RETURN Z=0 INVALID

00952                       * BYTE BUILDS 8 BIT VALUE FROM TWO HEX DIGITS IN
00953A FB75 8D   12     FB89 BYTE   BSR    BYTHEX   OBTAIN FIRST HEX
00954A FB77 C6   10        A        LDB    #16      PREPARE SHIFT
00955A FB79 3D                      MUL             OVER TO A
00956A FB7A 8D   0D     FB89        BSR    BYTHEX   OBTAIN SECOND HEX
00957A FB7C 34   04        A        PSHS   B        SAVE HIGH HEX
00958A FB7E AB   E0        A        ADDA   ,S+      COMBINE BOTH SIDES
00959A FB80 1F   89        A        TFR    A,B      SEND BACK IN B
00960A FB82 AB   62        A        ADDA   2,S      COMPUTE NEW CHECKSUM
00961A FB84 A7   62        A        STA    2,S      STORE BACK
00962A FB86 6A   63        A        DEC    3,S      DECREMENT BYTE COUNT
00963A FB88 39               BYTRTS RTS             RETURN TO CALLER

00965A FB89 3F               BYTHEX SWI             GET NEXT HEX
00966A FB8A      00        A        FCB    INCHNP   CHARACTER
00967A FB8B 17   01D4   FD62        LBSR   CNVHEX   CONVERT TO HEX
00968A FB8E 27   F8     FB88        BEQ    BYTRTS   RETURN IF VALID HEX
00969A FB90 35   F2        A        PULS   PC,U,Y,X,A RETURN TO CALLER WITH Z=0


00971                       * PUNCH STACK USE: S+8=TO ADDRESS
00972                       *                  S+6=RETURN ADDRESS
00973                       *                  S+4=SAVED PADDING VALUES
00974                       *                  S+2 FROM ADDRESS
00975                       *                  S+1=FRAME COUNT/CHECKSUM
00976                       *                  S+0=BYTE COUNT
00977A FB92 DE   F2        A BSDPUN LDU    VECTAB+.PAD LOAD PADDING VALUES
00978A FB94 AE   64        A        LDX    4,S      X=FROM ADDRESS
00979A FB96 34   56        A        PSHS   U,X,D    CREATE STACK WORK AREA
00980A FB98 CC   0018      A        LDD    #24      SET A=0, B=24
```

```
00981A FB9B D7    F2        A         STB    VECTAB+.PAD SETUP 24 CHARACTER PADS
00982A FB9D 3F                        SWI                SEND NULLS OUT
00983A FB9E       01        A         FCB    OUTCH       FUNCTION
00984A FB9F C6    04        A         LDB    #4          SETUP NEW LINE PAD TO 4
00985A FBA1 DD    F2        A         STD    VECTAB+.PAD SETUP PUNCH PADDING
00986                       * CALCULATE SIZE
00987A FBA3 EC    68        A BSPGO   LDD    8,S         LOAD TO
00988A FBA5 A3    62        A         SUBD   2,S         MINUS FROM=LENGTH
00989A FBA7 1083  0018      A         CMPD   #24         ? MORE THAN 23
00990A FBAB 25    02   FBAF           BLO    BSPOK       NO, OK
00991A FBAD C6    17        A         LDB    #23         FORCE TO 23 MAX
00992A FBAF 5C                BSPOK   INCB               PREPARE COUNTER
00993A FBB0 E7    E4        A         STB    ,S          STORE BYTE COUNT
00994A FBB2 CB    03        A         ADDB   #3          ADJUST TO FRAME COUNT
00995A FBB4 E7    61        A         STB    1,S         SAVE
00996                       *PUNCH CR,LF,NULS,S,1
00997A FBB6 30    8C 33               LEAX   <BSPSTR,PCR LOAD START RECORD HEADER
00998A FBB9 3F                        SWI                SEND OUT
00999A FBBA       03        A         FCB    PDATA       FUNCTION
01000                       * SEND FRAME COUNT
01001A FBBB 5F                        CLRB               INITIALIZE CHECKSUM
01002A FBBC 30    61        A         LEAX   1,S         POINT TO FRAME COUNT AND ADDR
01003A FBBE 8D    27   FBE7           BSR    BSPUN2      SEND FRAME COUNT
01004                       *DATA ADDRESS
01005A FBC0 8D    25   FBE7           BSR    BSPUN2      SEND ADDRESS HI
01006A FBC2 8D    23   FBE7           BSR    BSPUN2      SEND ADDRESS LOW
01007                       *PUNCH DATA
01008A FBC4 AE    62        A         LDX    2,S         LOAD START DATA ADDRESS
01009A FBC6 8D    1F   FBE7 BSPMRE BSR   BSPUN2      SEND OUT NEXT BYTE
01010A FBC8 6A    E4        A         DEC    ,S          ? FINAL BYTE
01011A FBCA 26    FA   FBC6           BNE    BSPMRE      LOOP IF NOT DONE
01012A FBCC AF    62        A         STX    2,S         UPDATE FROM ADDRESS VALUE
01013                       *PUNCH CHECKSUM
01014A FBCE 53                        COMB               COMPLEMENT
01015A FBCF E7    61        A         STB    1,S         STORE FOR SENDOUT
01016A FBD1 30    61        A         LEAX   1,S         POINT TO IT
01017A FBD3 8D    14   FBE9           BSR    BSPUNC      SEND OUT AS HEX
01018A FBD5 AE    68        A         LDX    8,S         LOAD TOP ADDRESS
01019A FBD7 AC    62        A         CMPX   2,S         ? DONE
01020A FBD9 24    C8   FBA3           BHS    BSPGO       BRANCH NOT
01021A FBDB 30    8C 11               LEAX   <BSPEOF,PCR PREPARE END OF FILE
01022A FBDE 3F                        SWI                SEND OUT STRING
01023A FBDF       03        A         FCB    PDATA       FUNCTION
01024A FBE0 EC    64        A         LDD    4,S         RECOVER PAD COUNTS
01025A FBE2 DD    F2        A         STD    VECTAB+.PAD RESTORE
01026A FBE4 4F                        CLRA               SET Z=1 FOR OK RETURN
01027A FBE5 35    D6        A         PULS   PC,U,X,D RETURN WITH OK CODE

01029A FBE7 EB    84        A BSPUN2  ADDB   ,X          ADD TO CHECKSUM
01030A FBE9 16    FDED F9D9 BSPUNC LBRA  ZOUT2H      SEND OUT AS HEX AND RETURN

01032A FBEC       53        A BSPSTR  FCB    'S,'1,EOT CR,LF,NULLS,S,1
01033A FBEF       53        A BSPEOF  FCC    /S9030000FC/EOF STRING
01034A FBF9       0D        A         FCB    CR,LF,EOT


01036                       * HSDTA - HIGH SPEED PRINT MEMORY
```

```
01037                          * INPUT: S+4=START ADDRESS
01038                          *        S+2=STOP ADDRESS
01039                          *        S+0=RETURN ADDRESS
01040                          * X,D VOLATILE

01042                          *  SEND TITLE
01043A FBFC 3F                 HSDTA  SWI                SEND NEW LINE
01044A FBFD      06        A          FCB     PCRLF      FUNCTION
01045A FBFE C6   06        A          LDB     #6         PREPARE 6 SPACES
01046A FC00 3F                 HSBLNK SWI                SEND BLANK
01047A FC01      07        A          FCB     SPACE      FUNCTION
01048A FC02 5A                        DECB               COUNT DOWN
01049A FC03 26   FB   FC00            BNE     HSBLNK     LOOP IF MORE
01050A FC05 5F                        CLRB               SETUP BYTE COUNT
01051A FC06 1F   98        A HSHTTL TFR     B,A        PREPARE FOR CONVERT
01052A FC08 17   FDDB F9E6            LBSR    ZOUTHX     CONVERT TO A HEX DIGIT
01053A FC0B 3F                        SWI                SEND BLANK
01054A FC0C      07        A          FCB     SPACE      FUNCTION
01055A FC0D 3F                        SWI                SEND ANOTHER
01056A FC0E      07        A          FCB     SPACE      BLANK
01057A FC0F 5C                        INCB               UP ANOTHER
01058A FC10 C1   10        A          CMPB    #$10       ? PAST 'F'
01059A FC12 25   F2   FC06            BLO     HSHTTL     LOOP UNTIL SO
01060A FC14 3F                 HSHLNE SWI                TO NEXT LINE
01061A FC15      06        A          FCB     PCRLF      FUNCTION
01062A FC16 25   2F   FC47            BCS     HSDRTN     RETURN IF USER ENTERED CTL-X
01063A FC18 30   64        A          LEAX    4,S        POINT AT ADDRESS TO CONVERT
01064A FC1A 3F                        SWI                PRINT OUT ADDRESS
01065A FC1B      05        A          FCB     OUT4HS     FUNCTION
01066A FC1C AE   64        A          LDX     4,S        LOAD ADDRESS PROPER
01067A FC1E C6   10        A          LDB     #16        NEXT SIXTEEN
01068A FC20 3F                 HSHNXT SWI                CONVERT BYTE TO HEX AND SEND
01069A FC21      04        A          FCB     OUT2HS     FUNCTION
01070A FC22 5A                        DECB               COUNT DOWN
01071A FC23 26   FB   FC20            BNE     HSHNXT     LOOP IF NOT SIXTEENTH
01072A FC25 3F                        SWI                SEND BLANK
01073A FC26      07        A          FCB     SPACE      FUNCTION
01074A FC27 AE   64        A          LDX     4,S        RELOAD FROM ADDRESS
01075A FC29 C6   10        A          LDB     #16        COUNT
01076A FC2B A6   80        A HSHCHR LDA     ,X+        NEXT BYTE
01077A FC2D 2B   04   FC33            BMI     HSHDOT     TOO LARGE, TO A DOT
01078A FC2F 81   20        A          CMPA    #'         ? LOWER THAN A BLANK
01079A FC31 24   02   FC35            BHS     HSHCOK     NO, BRANCH OK
01080A FC33 86   2E        A HSHDOT LDA     #'.        CONVERT INVALID TO A BLANK
01081A FC35 3F                 HSHCOK SWI                SEND CHARACTER
01082A FC36      01        A          FCB     OUTCH      FUNCTION
01083A FC37 5A                        DECB               ? DONE
01084A FC38 26   F1   FC2B            BNE     HSHCHR     BRANCH NO
01085A FC3A AC   62        A          CPX     2,S        ? PAST LAST ADDRESS
01086A FC3C 24   09   FC47            BHS     HSDRTN     QUIT IF SO
01087A FC3E AF   64        A          STX     4,S        UPDATE FROM ADDRESS
01088A FC40 A6   65        A          LDA     5,S        LOAD LOW BYTE ADDRESS
01089A FC42 48                        ASLA               ? TO SECTION BOUNDRY
01090A FC43 26   CF   FC14            BNE     HSHLNE     BRANCH IF NOT
01091A FC45 20   B5   FBFC            BRA     HSDTA      BRANCH IF SO
01092A FC47 3F                 HSDRTN SWI                SEND NEW LINE
01093A FC48      06        A          FCB     PCRLF      FUNCTION
01094A FC49 39                        RTS                RETURN TO CALLER
```

```
01095                         *F


01097                         ***********************************************
01098                         *     A S S I S T 0 9   C O M M A N D S
01099                         ***********************************************


01101                         *************REGISTERS - DISPLAY AND CHANGE REGISTERS
01102A FC4A 8D    23   FC6F CREG   BSR    REGPRT    PRINT REGISTERS
01103A FC4C 4C                     INCA             SET FOR CHANGE FUNCTION
01104A FC4D 8D    21   FC70        BSR    REGCHG    GO CHANGE, DISPLAY REGISTERS
01105A FC4F 39                     RTS              RETURN TO COMMAND PROCESSOR


01107                         *********************************************
01108                         *      REGPRT - PRINT/CHANGE REGISTERS SUBROUTINE
01109                         *  WILL ABORT TO 'CMDBAD' IF OVERFLOW DETECTED DURING
01110                         *  A CHANGE OPERATION.  CHANGE DISPLAYS REGISTERS WHEN
01111                         *  DONE.
01112                         * REGISTER MASK LIST CONSISTS OF:
01113                         *  A) CHARACTERS DENOTING REGISTER
01114                         *  B) ZERO FOR ONE BYTE, -1 FOR TWO
01115                         *  C) OFFSET ON STACK TO REGISTER POSITION
01116                         * INPUT: SP+4=STACKED REGISTERS
01117                         *         A=0 PRINT, A#0 PRINT AND CHANGE
01118                         * OUTPUT: (ONLY FOR REGISTER DISPLAY)
01119                         *          C=1 CONTROL-X ENTERED, C=0 OTHERWISE
01120                         * VOLATILE: D,X (CHANGE)
01121                         *            B,X (DISPLAY)
01122                         *********************************************
01123A FC50       50       A REGMSK FCB    'P,'C,-1,19 PC REG
01124A FC54       41       A        FCB    'A,0,10  A REG
01125A FC57       42       A        FCB    'B,0,11  B REG
01126A FC5A       58       A        FCB    'X,-1,13 X REG
01127A FC5D       59       A        FCB    'Y,-1,15 Y REG
01128A FC60       55       A        FCB    'U,-1,17 U REG
01129A FC63       53       A        FCB    'S,-1,1  S REG
01130A FC66       43       A        FCB    'C,'C,0,9 CC REG
01131A FC6A       44       A        FCB    'D,'P,0,12 DP REG
01132A FC6E       00       A        FCB    0          END OF LIST

01134A FC6F 4F               REGPRT CLRA              SETUP PRINT ONLY FLAG
01135A FC70 30    E8 10    A REGCHG LEAX   4+12,S     READY STACK VALUE
01136A FC73 34    32       A        PSHS   Y,X,A      SAVE ON STACK WITH OPTION
01137A FC75 31    8C D8             LEAY   REGMSK,PCR LOAD REGISTER MASK
01138A FC78 EC    A0       A REGP1  LDD    ,Y+        LOAD NEXT CHAR OR <=0
01139A FC7A 4D                      TSTA              ? END OF CHARACTERS
01140A FC7B 2F    04   FC81         BLE    REGP2      BRANCH NOT CHARACTER
01141A FC7D 3F                      SWI               SEND TO CONSOLE
01142A FC7E       01       A        FCB    OUTCH      FUNCTION BYTE
01143A FC7F 20    F7   FC78         BRA    REGP1      CHECK NEXT
01144A FC81 86    2D       A REGP2  LDA    #'-        READY '-'
01145A FC83 3F                      SWI               SEND OUT
01146A FC84       01       A        FCB    OUTCH      WITH OUTCH
01147A FC85 30    E5       A        LEAX   B,S        X->REGISTER TO PRINT
01148A FC87 6D    E4       A        TST    ,S         ? CHANGE OPTION
```

```
01149A FC89 26   12   FC9D          BNE   REGCNG     BRANCH YES
01150A FC8B 6D   3F      A          TST   -1,Y       ? ONE OR TWO BYTES
01151A FC8D 27   03   FC92          BEQ   REGP3      BRANCH ZERO MEANS ONE
01152A FC8F 3F                      SWI              PERFORM WORD HEX
01153A FC90      05      A          FCB   OUT4HS     FUNCTION
01154A FC91      8C      A          FCB   SKIP2      SKIP BYTE PRINT
01155A FC92 3F                REGP3 SWI              PERFORM BYTE HEX
01156A FC93      04      A          FCB   OUT2HS     FUNCTION
01157A FC94 EC   A0      A REG4     LDD   ,Y+        TO FRONT OF NEXT ENTRY
01158A FC96 5D                      TSTB             ? END OF ENTRIES
01159A FC97 26   DF   FC78          BNE   REGP1      LOOP IF MORE
01160A FC99 3F                      SWI              FORCE NEW LINE
01161A FC9A      06      A          FCB   PCRLF      FUNCTION
01162A FC9B 35   B2      A REGRTN   PULS  PC,Y,X,A   RESTORE STACK AND RETURN

01164A FC9D 8D   40   FCDF REGCNG   BSR   BLDNNB     INPUT BINARY NUMBER
01165A FC9F 27   10   FCB1          BEQ   REGNXC     IF CHANGE THEN JUMP
01166A FCA1 81   0D      A          CMPA  #CR        ? NO MORE DESIRED
01167A FCA3 27   1E   FCC3          BEQ   REGAGN     BRANCH NOPE
01168A FCA5 E6   3F      A          LDB   -1,Y       LOAD SIZE FLAG
01169A FCA7 5A                      DECB             MINUS ONE
01170A FCA8 50                      NEGB             MAKE POSITIVE
01171A FCA9 58                      ASLB             TIMES TWO (=2 OR =4)
01172A FCAA 3F                REGSKP SWI             PERFORM SPACES
01173A FCAB      07      A          FCB   SPACE      FUNCTION
01174A FCAC 5A                      DECB
01175A FCAD 26   FB   FCAA          BNE   REGSKP     LOOP IF MORE
01176A FCAF 20   E3   FC94          BRA   REG4       CONTINUE WITH NEXT REGISTER
01177A FCB1 A7   E4      A REGNXC   STA   ,S         SAVE DELIMITER IN OPTION
01178                      *                        (ALWAYS > 0)
01179A FCB3 DC   9B      A          LDD   NUMBER     OBTAIN BINARY RESULT
01180A FCB5 6D   3F      A          TST   -1,Y       ? TWO BYTES WORTH
01181A FCB7 26   02   FCBB          BNE   REGTWO     BRANCH YES
01182A FCB9 A6   82      A          LDA   ,-X        SETUP FOR TWO
01183A FCBB ED   84      A REGTWO   STD   ,X         STORE IN NEW VALUE
01184A FCBD A6   E4      A          LDA   ,S         RECOVER DELIMITER
01185A FCBF 81   0D      A          CMPA  #CR        ? END OF CHANGES
01186A FCC1 26   D1   FC94          BNE   REG4       NO, KEEP ON TRUCK'N
01187                      * MOVE STACKED DATA TO NEW STACK IN CASE STACK
01188                      * POINTER HAS CHANGED
01189A FCC3 30   8D E28A   REGAGN   LEAX  TSTACK,PCR LOAD TEMP AREA
01190A FCC7 C6   15      A          LDB   #21        LOAD COUNT
01191A FCC9 35   02      A REGTF1   PULS  A          NEXT BYTE
01192A FCCB A7   80      A          STA   ,X+        STORE INTO TEMP
01193A FCCD 5A                      DECB             COUNT DOWN
01194A FCCE 26   F9   FCC9          BNE   REGTF1     LOOP IF MORE
01195A FCD0 10EE 88 EC   A          LDS   -20,X      LOAD NEW STACK POINTER
01196A FCD4 C6   15      A          LDB   #21        LOAD COUNT AGAIN
01197A FCD6 A6   82      A REGTF2   LDA   ,-X        NEXT TO STORE
01198A FCD8 34   02      A          PSHS  A          BACK ONTO NEW STACK
01199A FCDA 5A                      DECB             COUNT DOWN
01200A FCDB 26   F9   FCD6          BNE   REGTF2     LOOP IF MORE
01201A FCDD 20   BC   FC9B          BRA   REGRTN     GO RESTART COMMAND

01203                      ********************************************
01204                      *  BLDNUM - BUILDS BINARY VALUE FROM INPUT HEX
01205                      *  THE ACTIVE EXPRESSION HANDLER IS USED.
```

```
01206                          * INPUT: S=RETURN ADDRESS
01207                          * OUTPUT: A=DELIMITER WHICH TERMINATED VALUE
01208                          *                           (IF DELM NOT ZERO)
01209                          *         "NUMBER"=WORD BINARY RESULT
01210                          *         Z=1 IF INPUT RECIEVED, Z=0 IF NO HEX RECIEVED
01211                          *  REGISTERS ARE TRANSPARENT
01212                          ***********************************************

01214                          * EXECUTE SINGLE OR EXTENDED ROM EXPRESSION HANDLER
01215                          *
01216                          * THE FLAG "DELIM" IS USED AS FOLLOWS:
01217                          *   DELIM=0  NO LEADING BLANKS, NO FORCED TERMINATOR
01218                          *   DELIM=CHR  ACCEPT LEADING 'CHR'S, FORCED TERMINATOR
01219A FCDF 4F                 BLDNNB CLRA               NO DYNAMIC DELIMITER
01220A FCE0      8C       A           FCB    SKIP2      SKIP NEXT INSTRUCTION
01221                          * BUILD WITH LEADING BLANKS
01222A FCE1 86   20       A BLDNUM LDA    #'         ALLOW LEADING BLANKS
01223A FCE3 97   8E       A           STA    DELIM      STORE AS DELIMITER
01224A FCE5 6E   9D E303              JMP    [VECTAB+.EXPAN,PCR] TO EXP ANALYZER

01226                          * THIS IS THE DEFAULT SINGLE ROM ANALYZER. WE ACCEPT:
01227                          *    1) HEX INPUT
01228                          *    2) 'M' FOR LAST MEMORY EXAMINE ADDRESS
01229                          *    3) 'P' FOR PROGRAM COUNTER ADDRESS
01230                          *    4) 'W' FOR WINDOW VALUE
01231                          *    5) '@' FOR INDIRECT VALUE
01232A FCE9 34   14       A EXP1   PSHS   X,B        SAVE REGISTERS
01233A FCEB 8D   5C    FD49 EXPDLM BSR    BLDHXI     CLEAR NUMBER, CHECK FIRST CHAR
01234A FCED 27   18    FD07        BEQ    EXP2       IF HEX DIGIT CONTINUE BUILDING
01235                          * SKIP BLANKS IF DESIRED
01236A FCEF 91   8E       A        CMPA   DELIM      ? CORRECT DELIMITER
01237A FCF1 27   F8    FCEB        BEQ    EXPDLM     YES, IGNORE IT
01238                          * TEST FOR M OR P
01239A FCF3 9E   9E       A        LDX    ADDR       DEFAULT FOR 'M'
01240A FCF5 81   4D       A        CMPA   #'M        ? MEMORY EXAMINE ADDR WANTED
01241A FCF7 27   16    FD0F        BEQ    EXPTDL     BRANCH IF SO
01242A FCF9 9E   93       A        LDX    PCNTER     DEFAULT FOR 'P'
01243A FCFB 81   50       A        CMPA   #'P        ? LAST PROGRAM COUNTER WANTED
01244A FCFD 27   10    FD0F        BEQ    EXPTDL     BRANCH IF SO
01245A FCFF 9E   A0       A        LDX    WINDOW     DEFAULT TO WINDOW
01246A FD01 81   57       A        CMPA   #'W        ? WINDOW WANTED
01247A FD03 27   0A    FD0F        BEQ    EXPTDL
01248A FD05 35   94       A EXPRTN PULS   PC,X,B     RETURN AND RESTORE REGISTERS
01249                          * GOT HEX, NOW CONTINUE BUILDING
01250A FD07 8D   44    FD4D EXP2   BSR    BLDHEX     COMPUTE NEXT DIGIT
01251A FD09 27   FC    FD07        BEQ    EXP2       CONTINUE IF MORE
01252A FD0B 20   0A    FD17        BRA    EXPCDL     SEARCH FOR +/-
01253                          * STORE VALUE AND CHECK IF NEED DELIMITER
01254A FD0D AE   84       A EXPTDI LDX    ,X         INDIRECTION DESIRED
01255A FD0F 9F   9B       A EXPTDL STX    NUMBER     STORE RESULT
01256A FD11 0D   8E       A        TST    DELIM      ? TO FORCE A DELIMITER
01257A FD13 27   F0    FD05        BEQ    EXPRTN     RETURN IF NOT WITH VALUE
01258A FD15 8D   62    FD79        BSR    READ       OBTAIN NEXT CHARACTER
01259                          * TEST FOR + OR -
01260A FD17 9E   9B       A EXPCDL LDX    NUMBER     LOAD LAST VALUE
01261A FD19 81   2B       A        CMPA   #'+        ? ADD OPERATOR
01262A FD1B 26   0E    FD2B        BNE    EXPCHM     BRANCH NOT
01263A FD1D 8D   23    FD42        BSR    EXPTRM     COMPUTE NEXT TERM
```

```
01264A FD1F 34   02        A        PSHS   A         SAVE DELIMITER
01265A FD21 DC   9B        A        LDD    NUMBER    LOAD NEW TERM
01266A FD23 30   8B        A EXPADD LEAX   D,X       ADD TO X
01267A FD25 9F   9B        A        STX    NUMBER    STORE AS NEW RESULT
01268A FD27 35   02        A        PULS   A         RESTORE DELIMITER
01269A FD29 20   EC   FD17          BRA    EXPCDL    NOW TEST IT
01270A FD2B 81   2D        A EXPCHM CMPA   #'-       ? SUBTRACT OPERATOR
01271A FD2D 27   07   FD36          BEQ    EXPSUB    BRANCH IF SO
01272A FD2F 81   40        A        CMPA   #'@       ? INDIRECTION DESIRED
01273A FD31 27   DA   FD0D          BEQ    EXPTDI    BRANCH IF SO
01274A FD33 5F                      CLRB             SET DELIMITER RETURN
01275A FD34 20   CF   FD05          BRA    EXPRTN    AND RETURN TO CALLER
01276A FD36 8D   0A   FD42 EXPSUB BSR    EXPTRM    OBTAIN NEXT TERM
01277A FD38 34   02        A        PSHS   A         SAVE DELIMITER
01278A FD3A DC   9B        A        LDD    NUMBER    LOAD UP NEXT TERM
01279A FD3C 40                      NEGA             NEGATE A
01280A FD3D 50                      NEGB             NEGATE B
01281A FD3E 82   00        A        SBCA   #0        CORRECT FOR A
01282A FD40 20   E1   FD23          BRA    EXPADD    GO ADD TO EXPRESION
01283                        * COMPUTE NEXT EXPRESSION TERM
01284                        * OUTPUT: X=OLD VALUE
01285                        *         'NUMBER'=NEXT TERM
01286A FD42 8D   9D   FCE1 EXPTRM BSR    BLDNUM    OBTAIN NEXT VALUE
01287A FD44 27   32   FD78          BEQ    CNVRTS    RETURN IF VALID NUMBER
01288A FD46 16   FC13 F95C BLDBAD LBRA   CMDBAD    ABORT COMMAND IF INVALID

01290                        *********************************************
01291                        *  BUILD BINARY VALUE USING INPUT CHARACTERS.
01292                        * INPUT: A=ASCII HEX VALUE OR DELIMITER
01293                        *        SP+0=RETURN ADDRESS
01294                        *        SP+2=16 BIT RESULT AREA
01295                        * OUTPUT: Z=1 A=BINARY VALUE
01296                        *         Z=0 IF INVALID HEX CHARACTER (A UNCHANGED)
01297                        * VOLATILE: D
01298                        ******************************************
01299A FD49 0F   9B        A BLDHXI CLR    NUMBER    CLEAR NUMBER
01300A FD4B 0F   9C        A        CLR    NUMBER+1  CLEAR NUMBER
01301A FD4D 8D   2A   FD79 BLDHEX BSR    READ      GET INPUT CHARACTER
01302A FD4F 8D   11   FD62 BLDHXC BSR    CNVHEX    CONVERT AND TEST CHARACTER
01303A FD51 26   25   FD78          BNE    CNVRTS    RETURN IF NOT A NUMBER
01304A FD53 C6   10        A        LDB    #16       PREPARE SHIFT
01305A FD55 3D                      MUL              BY FOUR PLACES
01306A FD56 86   04        A        LDA    #4        ROTATE BINARY INTO VALUE
01307A FD58 58               BLDSHF ASLB             OBTAIN NEXT BIT
01308A FD59 09   9C        A        ROL    NUMBER+1  INTO LOW BYTE
01309A FD5B 09   9B        A        ROL    NUMBER    INTO HI BYTE
01310A FD5D 4A                      DECA             COUNT DOWN
01311A FD5E 26   F8   FD58          BNE    BLDSHF    BRANCH IF MORE TO DO
01312A FD60 20   14   FD76          BRA    CNVOK     SET GOOD RETURN CODE

01314                        ******************************************
01315                        * CONVERT ASCII CHARACTER TO BINARY BYTE
01316                        * INPUT: A=ASCII
01317                        * OUTPUT: Z=1 A=BINARY VALUE
01318                        *         Z=0 IF INVALID
01319                        * ALL REGISTERS TRANSPARENT
```

```
01320                          * (A UNALTERED IF INVALID HEX)
01321                          ****************************************
01322A FD62 81   30      A CNVHEX CMPA   #'0      ? LOWER THAN A ZERO
01323A FD64 25   12   FD78        BLO    CNVRTS   BRANCH NOT VALUE
01324A FD66 81   39      A        CMPA   #'9      ? POSSIBLE A-F
01325A FD68 2F   0A   FD74        BLE    CNVGOT   BRANCH NO TO ACCEPT
01326A FD6A 81   41      A        CMPA   #'A      ? LESS THEN TEN
01327A FD6C 25   0A   FD78        BLO    CNVRTS   RETURN IF MINUS (INVALID)
01328A FD6E 81   46      A        CMPA   #'F      ? NOT TOO LARGE
01329A FD70 22   06   FD78        BHI    CNVRTS   NO, RETURN TOO LARGE
01330A FD72 80   07      A        SUBA   #7       DOWN TO BINARY
01331A FD74 84   0F      A CNVGOT ANDA   #$0F     CLEAR HIGH HEX
01332A FD76 1A   04      A CNVOK  ORCC   #4       FORCE ZERO ON FOR VALID HEX
01333A FD78 39             CNVRTS RTS             RETURN TO CALLER

01335                          * GET INPUT CHAR, ABORT COMMAND IF CONTROL-X (CANCEL)
01336A FD79 3F             READ   SWI             GET NEXT CHARACTER
01337A FD7A      00      A        FCB    INCHNP   FUNCTION
01338A FD7B 81   18      A        CMPA   #CAN     ? ABORT COMMAND
01339A FD7D 27   C7   FD46        BEQ    BLDBAD   BRANCH TO ABORT IF SO
01340A FD7F 39                    RTS             RETURN TO CALLER
01341                          *G

01343                          ***************GO - START PROGRAM EXECUTION
01344A FD80 8D   01   FD83 CGO    BSR    GOADDR   BUILD ADDRESS IF NEEDED
01345A FD82 3B                    RTI             START EXECUTING

01347                          * FIND OPTIONAL NEW PROGRAM COUNTER. ALSO ARM THE
01348                          * BREAKPOINTS.
01349A FD83 35   30      A GOADDR PULS   Y,X      RECOVER RETURN ADDRESS
01350A FD85 34   10      A        PSHS   X        STORE RETURN BACK
01351A FD87 26   19   FDA2        BNE    GONDFT   IF NO CARRIAGE RETURN THEN NEW PC
01352                          * DEFAULT PROGRAM COUNTER, SO FALL THROUGH IF
01353                          * IMMEDIATE BREAKPOINT.
01354A FD89 17   0186 FF42        LBSR   CBKLDR   SEARCH BREAKPOINTS
01355A FD8C AE   6C      A        LDX    12,S     LOAD PROGRAM COUNTER
01356A FD8E 5A             ARMBLP DECB            COUNT DOWN
01357A FD8F 2B   16   FDA7        BMI    ARMBK2   DONE, NONE TO SINGLE TRACE
01358A FD91 A6   30      A        LDA    -NUMBKP*2,Y PRE-FETCH OPCODE
01359A FD93 AC   A1      A        CMPX   ,Y++     ? IS THIS A BREAKPOINT
01360A FD95 26   F7   FD8E        BNE    ARMBLP   LOOP IF NOT
01361A FD97 81   3F      A        CMPA   #$3F     ? SWI BREAKPOINTED
01362A FD99 26   02   FD9D        BNE    ARMNSW   NO, SKIP SETTING OF PASS FLAG
01363A FD9B 97   FB      A        STA    SWIBFL   SHOW UPCOMMING SWI NOT BRKPNT
01364A FD9D 0C   8F      A ARMNSW INC    MISFLG   FLAG THRU A BREAKPOINT
01365A FD9F 16   0106 FEA8        LBRA   CDOT     DO SINGLE TRACE W/O BREAKPOINTS
01366                          * OBTAIN NEW PROGRAM COUNTER
01367A FDA2 17   00BB FE60 GONDFT LBSR   CDNUM    OBTAIN NEW PROGRAM COUNTER
01368A FDA5 ED   6C      A        STD    12,S     STORE INTO STACK
01369A FDA7 17   0198 FF42 ARMBK2 LBSR   CBKLDR   OBTAIN TABLE
01370A FDAA 00   FA      A        NEG    BKPTCT   COMPLEMENT TO SHOW ARMED
01371A FDAC 5A             ARMLOP DECB            ? DONE
01372A FDAD 2B   C9   FD78        BMI    CNVRTS   RETURN WHEN DONE
01373A FDAF A6   B4      A        LDA    [,Y]     LOAD OPCODE
01374A FDB1 A7   30      A        STA    -NUMBKP*2,Y STORE INTO OPCODE TABLE
```

```
01375A FDB3 86    3F        A         LDA    #$3F      READY "SWI" OPCODE
01376A FDB5 A7    B1        A         STA    [,Y++]    STORE AND MOVE UP TABLE
01377A FDB7 20    F3     FDAC         BRA    ARMLOP    AND CONTINUE

01379                          *******************CALL - CALL ADDRESS AS SUBROUTINE
01380A FDB9 8D    C8     FD83 CCALL   BSR    GOADDR    FETCH ADDRESS IF NEEDED
01381A FDBB 35    7F        A         PULS   U,Y,X,DP,D,CC RESTORE USERS REGISTERS
01382A FDBD AD    F1        A         JSR    [,S++]    CALL USER SUBROUTINE
01383A FDBF 3F                CGOBRK  SWI              PERFORM BREAKPOINT
01384A FDC0       0A        A         FCB    BRKPT     FUNCTION
01385A FDC1 20    FC     FDBF         BRA    CGOBRK    LOOP UNTIL USER CHANGES PC

01387                          ****************MEMORY - DISPLAY/CHANGE MEMORY
01388                          * CMEMN AND CMPADP ARE DIRECT ENTRY POINTS FROM
01389                          * THE COMMAND HANDLER FOR QUICK COMMANDS
01390A FDC3 17    009A FE60 CMEM      LBSR   CDNUM     OBTAIN ADDRESS
01391A FDC6 DD    9E        A CMEMN   STD    ADDR      STORE DEFAULT
01392A FDC8 9E    9E        A CMEM2   LDX    ADDR      LOAD POINTER
01393A FDCA 17    FC0C F9D9           LBSR   ZOUT2H    SEND OUT HEX VALUE OF BYTE
01394A FDCD 86    2D        A         LDA    #'-       LOAD DELIMITER
01395A FDCF 3F                        SWI              SEND OUT
01396A FDD0       01        A         FCB    OUTCH     FUNCTION
01397A FDD1 17    FF0B FCDF CMEM4     LBSR   BLDNNB    OBTAIN NEW BYTE VALUE
01398A FDD4 27    0A     FDE0         BEQ    CMENUM    BRANCH IF NUMBER
01399                          * COMA - SKIP BYTE
01400A FDD6 81    2C        A         CMPA   #',       ? COMMA
01401A FDD8 26    0E     FDE8         BNE    CMNOTC    BRANCH NOT
01402A FDDA 9F    9E        A         STX    ADDR      UPDATE POINTER
01403A FDDC 30    01        A         LEAX   1,X       TO NEXT BYTE
01404A FDDE 20    F1     FDD1         BRA    CMEM4     AND INPUT IT
01405A FDE0 D6    9C        A CMENUM  LDB    NUMBER+1  LOAD LOW BYTE VALUE
01406A FDE2 8D    47     FE2B         BSR    MUPDAT    GO OVERLAY MEMORY BYTE
01407A FDE4 81    2C        A         CMPA   #',       ? CONTINUE WITH NO DISPLAY
01408A FDE6 27    E9     FDD1         BEQ    CMEM4     BRANCH YES
01409                          * QUOTED STRING
01410A FDE8 81    27        A CMNOTC  CMPA   #''       ? QUOTED STRING
01411A FDEA 26    0C     FDF8         BNE    CMNOTQ    BRANCH NO
01412A FDEC 8D    8B     FD79 CMESTR  BSR    READ      OBTAIN NEXT CHARACTER
01413A FDEE 81    27        A         CMPA   #''       ? END OF QUOTED STRING
01414A FDF0 27    0C     FDFE         BEQ    CMSPCE    YES, QUIT STRING MODE
01415A FDF2 1F    89        A         TFR    A,B       TO B FOR SUBROUTINE
01416A FDF4 8D    35     FE2B         BSR    MUPDAT    GO UPDATE BYTE
01417A FDF6 20    F4     FDEC         BRA    CMESTR    GET NEXT CHARACTER
01418                          * BLANK - NEXT BYTE
01419A FDF8 81    20        A CMNOTQ  CMPA   #$20      ? BLANK FOR NEXT BYTE
01420A FDFA 26    06     FE02         BNE    CMNOTB    BRANCH NOT
01421A FDFC 9F    9E        A         STX    ADDR      UPDATE POINTER
01422A FDFE 3F                CMSPCE  SWI              GIVE SPACE
01423A FDFF       07        A         FCB    SPACE     FUNCTION
01424A FE00 20    C6     FDC8         BRA    CMEM2     NOW PROMPT FOR NEXT
01425                          * LINE FEED - NEXT BYTE WITH ADDRESS
01426A FE02 81    0A        A CMNOTB  CMPA   #LF       ? LINE FEED FOR NEXT BYTE
01427A FE04 26    08     FE0E         BNE    CMNOTL    BRANCH NO
01428A FE06 86    0D        A         LDA    #CR       GIVE CARRIAGE RETURN
```

```
01429A FE08 3F                      SWI                TO CONSOLE
01430A FE09      01       A         FCB     OUTCH      HANDLER
01431A FE0A 9F   9E       A         STX     ADDR       STORE NEXT ADDRESS
01432A FE0C 20   0A    FE18         BRA     CMPADP     BRANCH TO SHOW
01433                       * UP ARROW - PREVIOUS BYTE AND ADDRESS
01434A FE0E 81   5E       A CMNOTL CMPA     #'@        ? UP ARROW FOR PREVIOUS BYTE
01435A FE10 26   0A    FE1C         BNE     CMNOTU     BRANCH NOT
01436A FE12 30   1E       A         LEAX    -2,X       DOWN TO PREVIOUS BYTE
01437A FE14 9F   9E       A         STX     ADDR       STORE NEW POINTER
01438A FE16 3F              CMPADS SWI                 FORCE NEW LINE
01439A FE17      06       A         FCB     PCRLF      FUNCTION
01440A FE18 8D   07    FE21 CMPADP BSR      PRTADR     GO PRINT ITS VALUE
01441A FE1A 20   AC    FDC8         BRA     CMEM2      THEN PROMPT FOR INPUT
01442                       * SLASH - NEXT BYTE WITH ADDRESS
01443A FE1C 81   2F       A CMNOTU CMPA     #'/        ? SLASH FOR CURRENT DISPLAY
01444A FE1E 27   F6    FE16         BEQ     CMPADS     YES, SEND ADDRESS
01445A FE20 39                      RTS                RETURN FROM COMMAND

01447                       * PRINT CURRENT ADDRESS
01448A FE21 9E   9E       A PRTADR LDX      ADDR       LOAD POINTER VALUE
01449A FE23 34   10       A         PSHS    X          SAVE X ON STACK
01450A FE25 30   E4       A         LEAX    ,S         POINT TO IT FOR DISPLAY
01451A FE27 3F                      SWI                DISPLAY POINTER IN HEX
01452A FE28      05       A         FCB     OUT4HS     FUNCTION
01453A FE29 35   90       A         PULS    PC,X       RECOVER POINTER AND RETURN

01455                       * UPDATE BYTE
01456A FE2B 9E   9E       A MUPDAT LDX      ADDR       LOAD NEXT BYTE POINTER
01457A FE2D E7   80       A         STB     ,X+        STORE AND INCREMENT X
01458A FE2F E1   1F       A         CMPB    -1,X       ? SUCCESFULL STORE
01459A FE31 26   03    FE36         BNE     MUPBAD     BRANCH FOR '?' IF NOT
01460A FE33 9F   9E       A         STX     ADDR       STORE NEW POINTER VALUE
01461A FE35 39                      RTS                BACK TO CALLER
01462A FE36 34   02       A MUPBAD PSHS     A          SAVE A REGISTER
01463A FE38 86   3F       A         LDA     #'?        SHOW INVALID
01464A FE3A 3F                      SWI                SEND OUT
01465A FE3B      01       A         FCB     OUTCH      FUNCTION
01466A FE3C 35   82       A         PULS    PC,A       RETURN TO CALLER


01468                       ********************WINDOW  -  SET WINDOW VALUE
01469A FE3E 8D   20    FE60 CWINDO BSR      CDNUM      OBTAIN WINDOW VALUE
01470A FE40 DD   A0       A         STD     WINDOW     STORE IT IN
01471A FE42 39                      RTS                END COMMAND


01473                       ******************DISPLAY - HIGH SPEED DISPLAY MEMORY
01474A FE43 8D   1B    FE60 CDISP  BSR      CDNUM      FETCH ADDRESS
01475A FE45 C4   F0       A         ANDB    #$F0       FORCE TO 16 BOUNDRY
01476A FE47 1F   02       A         TFR     D,Y        SAVE IN Y
01477A FE49 30   2F       A         LEAX    15,Y       DEFAULT LENGTH
01478A FE4B 25   04    FE51         BCS     CDISPS     BRANCH IF END OF INPUT
01479A FE4D 8D   11    FE60         BSR     CDNUM      OBTAIN COUNT
01480A FE4F 30   AB       A         LEAX    D,Y        ASSUME COUNT, COMPUTE END ADDR
01481A FE51 34   30       A CDISPS PSHS     Y,X        SETUP PARAMETERS FOR HSDATA
01482A FE53 10A3 62       A         CMPD    2,S        ? WAS IT COUNT
```

```
01483A FE56 23    02   FE5A          BLS    CDCNT     BRANCH YES
01484A FE58 ED    E4      A          STD    ,S        STORE HIGH ADDRESS
01485A FE5A AD    9D E184     CDCNT  JSR    [VECTAB+.HSDTA,PCR] CALL PRINT ROUTINE
01486A FE5E 35    E0      A          PULS   PC,U,Y    CLEAN STACK AND END COMMAND

01488                         * OBTAIN NUMBER - ABORT IF NONE
01489                         * ONLY DELIMITERS OF CR, BLANK, OR '/' ARE ACCEPTED
01490                         * OUTPUT: D=VALUE, C=1 IF CARRIAGE RETURN DELMITER,
01491                         *                                   ELSE C=0
01492A FE60 17    FE7E FCE1 CDNUM  LBSR   BLDNUM    OBTAIN NUMBER
01493A FE63 26    09   FE6E          BNE    CDBADN    BRANCH IF INVALID
01494A FE65 81    2F      A          CMPA   #'/       ? VALID DELIMITER
01495A FE67 22    05   FE6E          BHI    CDBADN    BRANCH IF NOT FOR ERROR
01496A FE69 81    0E      A          CMPA   #CR+1     LEAVE COMPARE FOR CARRIAGE RET
01497A FE6B DC    9B      A          LDD    NUMBER    LOAD NUMBER
01498A FE6D 39                       RTS              RETURN WITH COMPARE
01499A FE6E 16    FAEB F95C CDBADN LBRA   CMDBAD    RETURN TO ERROR MECHANISM


01501                         *****************PUNCH - PUNCH MEMORY IN S1-S9 FORMAT
01502A FE71 8D    ED   FE60 CPUNCH BSR    CDNUM     OBTAIN START ADDRESS
01503A FE73 1F    02      A          TFR    D,Y       SAVE IN Y
01504A FE75 8D    E9   FE60          BSR    CDNUM     OBTAIN END ADDRESS
01505A FE77 6F    E2      A          CLR    ,-S       SETUP PUNCH FUNCTION CODE
01506A FE79 34    26      A          PSHS   Y,D       STORE VALUES ON STACK
01507A FE7B AD    9D E165   CCALBS JSR    [VECTAB+.BSON,PCR] INITIALIZE HANDLER
01508A FE7F AD    9D E163          JSR    [VECTAB+.BSDTA,PCR] PERFORM FUNCTION
01509A FE83 34    01      A          PSHS   CC        SAVE RETURN CODE
01510A FE85 AD    9D E15F          JSR    [VECTAB+.BSOFF,PCR] TURN OFF HANDLER
01511A FE89 35    01      A          PULS   CC        OBTAIN CONDITION CODE SAVED
01512A FE8B 26    E1   FE6E          BNE    CDBADN    BRANCH IF ERROR
01513A FE8D 35    B2      A          PULS   PC,Y,X,A RETURN FROM COMMAND


01515                         *****************LOAD - LOAD MEMORY FROM S1-S9 FORMAT
01516A FE8F 8D    01   FE92 CLOAD  BSR    CLVOFS    CALL SETUP AND PASS CODE
01517A FE91       01      A          FCB    1         LOAD FUNCTION CODE FOR PACKET

01519A FE92 33    F1      A CLVOFS LEAU   [,S++]    LOAD CODE IN HIGH BYTE OF U
01520A FE94 33    D4      A          LEAU   [,U]      NOT CHANGING CC AND RESTORE S
01521A FE96 27    03   FE9B          BEQ    CLVDFT    BRANCH IF CARRIAGE RETURN NEXT
01522A FE98 8D    C6   FE60          BSR    CDNUM     OBTAIN OFFSET
01523A FE9A       8C      A          FCB    SKIP2     SKIP DEFAULT OFFSET
01524A FE9B 4F              CLVDFT CLRA             CREATE ZERO OFFSET
01525A FE9C 5F                       CLRB             AS DEFAULT
01526A FE9D 34    4E      A          PSHS   U,DP,D    SETUP CODE, NULL WORD, OFFSET
01527A FE9F 20    DA   FE7B          BRA    CCALBS    ENTER CALL TO BS ROUTINES


01529                         ******************VERIFY - COMPARE MEMORY WITH FILES
01530A FEA1 8D    EF   FE92 CVER   BSR    CLVOFS    COMPUTE OFFSET IF ANY
01531A FEA3       FF      A          FCB    -1        VERIFY FNCTN CODE FOR PACKET
```

```
01533                          *******************TRACE - TRACE INSTRUCTIONS
01534                          ******************* . - SINGLE STEP TRACE
01535A FEA4 8D    BA    FE60 CTRACE BSR    CDNUM    OBTAIN TRACE COUNT
01536A FEA6 DD    91       A        STD    TRACEC   STORE COUNT
01537A FEA8 32    62       A CDOT   LEAS   2,S      RID COMMAND RETURN FROM STACK
01538A FEAA EE    F8 0A    A CTRCE3 LDU    [10,S]   LOAD OPCODE TO EXECUTE
01539A FEAD DF    99       A        STU    LASTOP   STORE FOR TRACE INTERRUPT
01540A FEAF DE    F6       A        LDU    VECTAB+.PTM LOAD PTM ADDRESS
01541A FEB1 CC    0701     A        LDD    #7!<8+1  CYCLES DOWN+CYCLES UP
01542A FEB4 ED    42       A        STD    PTMTM1-PTM,U START NMI TIMEOUT
01543A FEB6 3B                      RTI             RETURN FOR ONE INSTRUCTION


01545                          *************NULLS  -  SET NEW LINE AND CHAR PADDING
01546A FEB7 8D    A7    FE60 CNULLS BSR.   CDNUM    OBTAIN NEW LINE PAD
01547A FEB9 DD    F2       A        STD    VECTAB+.PAD RESET VALUES
01548A FEBB 39                      RTS             END COMMAND


01550                          ******************STLEVEL - SET STACK TRACE LEVEL
01551A FEBC 27    05    FEC3 CSTLEV BEQ    STLDFT   TAKE DEFAULT
01552A FEBE 8D    A0    FE60        BSR    CDNUM    OBTAIN NEW STACK LEVEL
01553A FEC0 DD    F8       A        STD    SLEVEL   STORE NEW ENTRY
01554A FEC2 39                      RTS             TO COMMAND HANDLER
01555A FEC3 30    6E       A STLDFT LEAX   14,S     COMPUTE NMI COMPARE
01556A FEC5 9F    F8       A        STX    SLEVEL   AND STORE IT
01557A FEC7 39                      RTS             END COMMAND


01559                          ******************OFFSET - COMPUTE SHORT AND LONG
01560                          ******************                BRANCH OFFSETS
01561A FEC8 8D    96    FE60 COFFS  BSR    CDNUM    OBTAIN INSTRUCTION ADDRESS
01562A FECA 1F    01       A        TFR    D,X      USE AS FROM ADDRESS
01563A FECC 8D    92    FE60        BSR    CDNUM    OBTAIN TO ADDRESS
01564                          * D=TO INSTRUCTION, X=FROM INSTRUCTION OFFSET BYTE(S)
01565A FECE 30    01       A        LEAX   1,X      ADJUST FOR *+2 SHORT BRANCH
01566A FED0 34    30       A        PSHS   Y,X      STORE WORK WORD AND VALUE ON S
01567A FED2 A3    E4       A        SUBD   ,S       FIND OFFSET
01568A FED4 ED    E4       A        STD    ,S       SAVE OVER STACK
01569A FED6 30    61       A        LEAX   1,S      POINT FOR ONE BYTE DISPLAY
01570A FED8 1D                      SEX             SIGN EXTEND LOW BYTE
01571A FED9 A1    E4       A        CMPA   ,S       ? VALID ONE BYTE OFFSET
01572A FEDB 26    02    FEDF        BNE    COFNO1   BRANCH IF NOT
01573A FEDD 3F                      SWI             SHOW ONE BYTE OFFSET
01574A FEDE       04       A        FCB    OUT2HS   FUNCTION
01575A FEDF EE    E4       A COFNO1 LDU    ,S       RELOAD OFFSET
01576A FEE1 33    5F       A        LEAU   -1,U     CONVERT TO LONG BRANCH OFFSET
01577A FEE3 EF    84       A        STU    ,X       STORE BACK WHERE X POINTS NOW
01578A FEE5 3F                      SWI             SHOW TWO BYTE OFFSET
01579A FEE6       05       A        FCB    OUT4HS   FUNCTION
01580A FEE7 3F                      SWI             FORCE NEW LINE
01581A FEE8       06       A        FCB    PCRLF    FUNCTION
01582A FEE9 35    96       A        PULS   PC,X,D   RESTORE STACK AND END COMMAND
01583                          *H
```

```
01585                          **************BREAKPOINT - DISPLAY/ENTER/DELETE/CLEAR
01586                          **************            BREAKPOINTS
01587A FEEB 27    23   FF10 CBKPT  BEQ    CBKDSP   BRANCH DISPLAY OF JUST 'B'
01588A FEED 17    FDF1 FCE1        LBSR   BLDNUM   ATTEMPT VALUE ENTRY
01589A FEF0 27    2C   FF1E        BEQ    CBKADD   BRANCH TO ADD IF SO
01590A FEF2 81    2D      A        CMPA   #'-      ? CORRECT DELIMITER
01591A FEF4 26    3F   FF35        BNE    CBKERR   NO, BRANCH FOR ERROR
01592A FEF6 17    FDE8 FCE1        LBSR   BLDNUM   ATTEMPT DELETE VALUE
01593A FEF9 27    03   FEFE        BEQ    CBKDLE   GOT ONE, GO DELETE IT
01594A FEFB 0F    FA      A        CLR    BKPTCT   WAS 'B -', SO ZERO COUNT
01595A FEFD 39              CBKRTS RTS             END COMMAND
01596                       * DELETE THE ENTRY
01597A FEFE 8D    40   FF40 CBKDLE BSR    CBKSET   SETUP REGISTERS AND VALUE
01598A FF00 5A              CBKDLP DECB            ? ANY ENTRIES IN TABLE
01599A FF01 2B    32   FF35        BMI    CBKERR   BRANCH NO, ERROR
01600A FF03 AC    A1      A        CMPX   ,Y++     ? IS THIS THE ENTRY
01601A FF05 26    F9   FF00        BNE    CBKDLP   NO, TRY NEXT
01602                       * FOUND, NOW MOVE OTHERS UP IN ITS PLACE
01603A FF07 AE    A1      A CBKDLM LDX    ,Y++     LOAD NEXT ONE UP
01604A FF09 AF    3C      A        STX    -4,Y     MOVE DOWN BY ONE
01605A FF0B 5A                     DECB            ? DONE
01606A FF0C 2A    F9   FF07        BPL    CBKDLM   NO, CONTINUE MOVE
01607A FF0E 0A    FA      A        DEC    BKPTCT   DECREMENT BREAKPOINT COUNT
01608A FF10 8D    2E   FF40 CBKDSP BSR    CBKSET   SETUP REGISTERS AND LOAD VALUE
01609A FF12 27    E9   FEFD        BEQ    CBKRTS   RETURN IF NONE TO DISPLY
01610A FF14 30    A1      A CBKDSL LEAX   ,Y++     POINT TO NEXT ENTRY
01611A FF16 3F                     SWI             DISPLAY IN HEX
01612A FF17       05      A        FCB    OUT4HS   FUNCTION
01613A FF18 5A                     DECB            COUNT DOWN
01614A FF19 26    F9   FF14        BNE    CBKDSL   LOOP IF MORE TO DO
01615A FF1B 3F                     SWI             SKIP TO NEW LINE
01616A FF1C       06      A        FCB    PCRLF    FUNCTION
01617A FF1D 39                     RTS             RETURN TO END COMMAND
01618                       * ADD NEW ENTRY
01619A FF1E 8D    20   FF40 CBKADD BSR    CBKSET   SETUP REGISTERS
01620A FF20 C1    08      A        CMPB   #NUMBKP  ? ALREADY FULL
01621A FF22 27    11   FF35        BEQ    CBKERR   BRANCH ERROR IF SO
01622A FF24 A6    84      A        LDA    ,X       LOAD BYTE TO TRAP
01623A FF26 E7    84      A        STB    ,X       TRY TO CHANGE
01624A FF28 E1    84      A        CMPB   ,X       ? CHANGABLE RAM
01625A FF2A 26    09   FF35        BNE    CBKERR   BRANCH ERROR IF NOT
01626A FF2C A7    84      A        STA    ,X       RESTORE BYTE
01627A FF2E 5A              CBKADL DECB            COUNT DOWN
01628A FF2F 2B    07   FF38        BMI    CBKADT   BRANCH IF DONE TO ADD IT
01629A FF31 AC    A1      A        CMPX   ,Y++     ? ENTRY ALREADY HERE
01630A FF33 26    F9   FF2E        BNE    CBKADL   LOOP IF NOT
01631A FF35 16    FA24 F95C CBKERR LBRA   CMDBAD   RETURN TO ERROR PRODUCE
01632A FF38 AF    A4      A CBKADT STX    ,Y       ADD THIS ENTRY
01633A FF3A 6F    31      A        CLR    -NUMBKP*2+1,Y CLEAR OPTIONAL BYTE
01634A FF3C 0C    FA      A        INC    BKPTCT   ADD ONE TO COUNT
01635A FF3E 20    D0   FF10        BRA    CBKDSP   AND NOW DISPLAY ALL OF 'EM
01636                       * SETUP REGISTERS FOR SCAN
01637A FF40 9E    9B      A CBKSET LDX    NUMBER   LOAD VALUE DESIRED
01638A FF42 31    8D E06C   CBKLDR LEAY   BKPTBL,PCR LOAD START OF TABLE
01639A FF46 D6    FA      A        LDB    BKPTCT   LOAD ENTRY COUNT
01640A FF48 39                     RTS             RETURN
```

```
01642                              *****************ENCODE  -  ENCODE A POSTBYTE
01643A FF49 6F    E2        A CENCDE CLR    ,-S        DEFAULT TO NOT INDIRECT
01644A FF4B 5F                       CLRB              ZERO POSTBYTE VALUE
01645A FF4C 30    8C 3F              LEAX   <CONV1,PCR START TABLE SEARCH
01646A FF4F 3F                       SWI               OBTAIN FIRST CHARACTER
01647A FF50       00        A        FCB    INCHNP     FUNCTION
01648A FF51 81    5B        A        CMPA   #'[        ? INDIRECT HERE
01649A FF53 26    06   FF5B          BNE    CEN2       BRANCH IF NOT
01650A FF55 86    10        A        LDA    #$10       SET INDIRECT BIT ON
01651A FF57 A7    E4        A        STA    ,S         SAVE FOR LATER
01652A FF59 3F                CENGET SWI               OBTAIN NEXT CHARACTER
01653A FF5A       00        A        FCB    INCHNP     FUNCTION
01654A FF5B 81    0D        A CEN2   CMPA   #CR        ? END OF ENTRY
01655A FF5D 27    0C   FF6B          BEQ    CEND1      BRANCH YES
01656A FF5F 6D    84        A CENLP1 TST    ,X         ? END OF TABLE
01657A FF61 2B    D2   FF35          BMI    CBKERR     BRANCH ERROR IF SO
01658A FF63 A1    81        A        CMPA   ,X++       ? THIS THE CHARACTER
01659A FF65 26    F8   FF5F          BNE    CENLP1     BRANCH IF NOT
01660A FF67 EB    1F        A        ADDB   -1,X       ADD THIS VALUE
01661A FF69 20    EE   FF59          BRA    CENGET     GET NEXT INPUT
01662A FF6B 30    8C 49       CEND1  LEAX   <CONV2,PCR POINT AT TABLE 2
01663A FF6E 1F    98        A        TFR    B,A        SAVE COPY IN A
01664A FF70 84    60        A        ANDA   #$60       ISOLATE REGISTER MASK
01665A FF72 AA    E4        A        ORA    ,S         ADD IN INDIRECTION BIT
01666A FF74 A7    E4        A        STA    ,S         SAVE BACK AS POSTBYTE SKELETON
01667A FF76 C4    9F        A        ANDB   #$9F       CLEAR REGISTER BITS
01668A FF78 6D    84        A CENLP2 TST    ,X         ? END OF TABLE
01669A FF7A 27    B9   FF35          BEQ    CBKERR     BRANCH ERROR IF SO
01670A FF7C E1    81        A        CMPB   ,X++       ? SAME VALUE
01671A FF7E 26    F8   FF78          BNE    CENLP2     LOOP IF NOT
01672A FF80 E6    1F        A        LDB    -1,X       LOAD RESULT VALUE
01673A FF82 EA    E4        A        ORB    ,S         ADD TO BASE SKELETON
01674A FF84 E7    E4        A        STB    ,S         SAVE POSTBYTE ON STACK
01675A FF86 30    E4        A        LEAX   ,S         POINT TO IT
01676A FF88 3F                       SWI               SEND OUT AS HEX
01677A FF89       04        A        FCB    OUT2HS     FUNCTION
01678A FF8A 3F                       SWI               TO NEXT LINE
01679A FF8B       06        A        FCB    PCRLF      FUNCTION
01680A FF8C 35    84        A        PULS   PC,B       END OF COMMAND

01682                              * TABLE ONE DEFINES VALID INPUT IN SEQUENCE
01683A FF8E       41        A CONV1  FCB    'A,$04,'B,$05,'D,$06,'H,$01
01684A FF96       48        A        FCB    'H,$01,'H,$01,'H,$00,',,$00
01685A FF9E       2D        A        FCB    '-,$09,'-,$01,'S,$70,'Y,$30
01686A FFA6       55        A        FCB    'U,$50,'X,$10,'+,$07,'+,$01
01687A FFAE       50        A        FCB    'P,$80,'C,$00,'R,$00,'],$00
01688A FFB6       FF        A        FCB    $FF        END OF TABLE
01689                              *CONV2 USES ABOVE CONVERSION TO SET POSTBYTE
01690                              *                               BIT SKELETON.
01691A FFB7       1084      A CONV2  FDB    $1084,$1100 R,         H,R
01692A FFBB       1288      A        FDB    $1288,$1389 HH,R       HHHH,R
01693A FFBF       1486      A        FDB    $1486,$1585 A,R        B,R
01694A FFC3       168B      A        FDB    $168B,$1780 D,R        ,R+
01695A FFC7       1881      A        FDB    $1881,$1982 ,R++       ,-R
01696A FFCB       1A83      A        FDB    $1A83,$828C ,--R       HH,PCR
01697A FFCF       838D      A        FDB    $838D,$039F HHHH,PCR   [HHHH]
01698A FFD3       00        A        FCB    0          END OF TABLE
```

```
01700                               *****************************************************
01701                               *              DEFAULT INTERRUPT TRANSFERS          *
01702                               *****************************************************
01703A FFD4 6E    9D DFEE   RSRVD  JMP    [VECTAB+.RSVD,PCR] RESERVED VECTOR
01704A FFD8 6E    9D DFEC   SWI3   JMP    [VECTAB+.SWI3,PCR] SWI3 VECTOR
01705A FFDC 6E    9D DFEA   SWI2   JMP    [VECTAB+.SWI2,PCR] SWI2 VECTOR
01706A FFE0 6E    9D DFE8   FIRQ   JMP    [VECTAB+.FIRQ,PCR] FIRQ VECTOR
01707A FFE4 6E    9D DFE6   IRQ    JMP    [VECTAB+.IRQ,PCR] IRQ VECTOR
01708A FFE8 6E    9D DFE4   SWI    JMP    [VECTAB+.SWI,PCR] SWI VECTOR
01709A FFEC 6E    9D DFE2   NMI    JMP    [VECTAB+.NMI,PCR] NMI VECTOR


01711                               *******************************************************
01712                               *              ASSIST09 HARDWARE VECTOR TABLE
01713                               *  THIS TABLE IS USED IF THE ASSIST09 ROM ADDRESSES
01714                               *  THE MC6809 HARDWARE VECTORS.
01715                               *******************************************************
01716A FFF0                                ORG    ROMBEG+ROMSIZ-16 SETUP HARDWARE VECTORS
01717A FFF0       FFD4     A               FDB    RSRVD     RESERVED SLOT
01718A FFF2       FFD8     A               FDB    SWI3      SOFTWARE INTERRUPT 3
01719A FFF4       FFDC     A               FDB    SWI2      SOFTWARE INTERRUPT 2
01720A FFF6       FFE0     A               FDB    FIRQ      FAST INTERRUPT REQUEST
01721A FFF8       FFE4     A               FDB    IRQ       INTERRUPT REQUEST
01722A FFFA       FFE8     A               FDB    SWI       SOFTWARE INTERRUPT
01723A FFFC       FFEC     A               FDB    NMI       NON-MASKABLE INTERRUPT
01724A FFFE       F837     A               FDB    RESET     RESTART


01726             F837     A               END    RESET
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000


   002E .ACIA  00095*00825 00837 00853
   0000 .AVTBL 00072*00594
   0024 .BSDTA 00090*01508
   0026 .BSOFF 00091*01510
   0022 .BSON  00089*01507
   0016 .CIDTA 00083*00725
   0018 .CIOFF 00084*
   0014 .CION  00082*00348
   0002 .CMDL1 00073*00429
   002C .CMDL2 00094*00432
   001C .CODTA 00086*00568
   001E .COOFF 00087*
   001A .COON  00085*00349
   0032 .ECHO  00097*00625
   002A .EXPAN 00093*01224
   000A .FIRQ  00077*01706
   0020 .HSDTA 00088*01485
   000C .IRQ   00078*01707
   0010 .NMI   00080*01709
   0030 .PAD   00096*00857 00860 00977 00981 00985 01025 01547
   0028 .PAUSE 00092*00724
   0034 .PTM   00098*00353 01540
```

```
0012 .RESET 00081*
0004 .RSVD  00074*01703
000E .SWI   00079*01708
0008 .SWI2  00076*01705
0006 .SWI3  00075*01704
E008 ACIA   00024*00256
DF9E ADDR   00133*01239 01391 01392 01402 01421 01431 01437 01448 01456 01460
FDA7 ARMBK2 00773 01357 01369*
FD8E ARMBLP 01356*01360
FDAC ARMLOP 01371*01377
FD9D ARMNSW 01362 01364*
DF9D BASEPG 00135*00186 00784
0007 BELL   00036*00782
DFB2 BKPTBL 00127*01638
DFFA BKPTCT 00121*00386 01370 01594 01607 01634 01639
DFA2 BKPTOP 00129*
F815 BLD2   00192*00196
F821 BLD3   00198*00201
FD46 BLDBAD 01288*01339
FD4D BLDHEX 01250 01301*
FD4F BLDHXC 00421 01302*
FD49 BLDHXI 01233 01299*
FCDF BLDNNB 01164 01219*01397
FCE1 BLDNUM 01222*01286 01492 01588 01592
F835 BLDRTN 00205 00207*
FD58 BLDSHF 01307*01311
F800 BLDVTR 00183*00218
000A BRKPT  00066*01384
FB6A BSDCMP 00942 00944*
FB70 BSDEOL 00940 00948*
FB40 BSDLD1 00919*00922 00949
FB42 BSDLD2 00921*00928
FB60 BSDNXT 00939*00945
FB92 BSDPUN 00913 00977*
FB6E BSDSRT 00926 00946*00950
FB38 BSDTA  00250 00911*
FB27 BSOFF  00251 00891*
FB33 BSOFLP 00899*00900
FB1B BSON   00249 00880*
FB22 BSON2  00882 00884*
FBEF BSPEOF 01021 01033*
FBA3 BSPGO  00987*01020
FBC6 BSPMRE 01009*01011
FBAF BSPOK  00990 00992*
FBEC BSPSTR 00997 01032*
FBE7 BSPUN2 01003 01005 01006 01009 01029*
FBE9 BSPUNC 01017 01030*
FB75 BYTE   00930 00933 00935 00939 00953*
FB89 BYTHEX 00953 00956 00965*
FB88 BYTRTS 00963*00968
0018 CAN    00040*00711 00718 01338
FF1E CBKADD 01589 01619*
FF2E CBKADL 01627*01630
FF38 CBKADT 01628 01632*
FEFE CBKDLE 01593 01597*
FF07 CBKDLM 01603*01606
FF00 CBKDLP 01598*01601
FF14 CBKDSL 01610*01614
```

```
FF10 CBKDSP 01587 01608*01635
FF35 CBKERR 01591 01599 01621 01625 01631*01657 01669
FF42 CBKLDR 00303 00383 01354 01369 01638*
FEEB CBKPT  00503 01587*
FEFD CBKRTS 01595*01609
FF40 CBKSET 01597 01608 01619 01637*
FE7B CCALBS 01507*01527
FDB9 CCALL  00506 01380*
FE6E CDBADN 01493 01495 01499*01512
FE5A CDCNT  01483 01485*
FE43 CDISP  00509 01474*
FE51 CDISPS 01478 01481*
FE60 CDNUM  01367 01390 01469 01474 01479 01492*01502 01504 01522 01535 01546
            01552 01561 01563
FEA8 CDOT   00408 01365 01537*
FF5B CEN2   01649 01654*
FF49 CENCDE 00512 01643*
FF6B CEND1  01655 01662*
FF59 CENGET 01652*01661
FF5F CENLP1 01656*01659
FF78 CENLP2 01668*01671
FD80 CGO    00515 01344*
FDBF CGOBRK 01383*01385
FA58 CHKABT 00701 00709*00764
FA61 CHKRTN 00710 00714*
FA60 CHKSEC 00713*00719
FA62 CHKWT  00712 00715*00717
FADC CIDTA  00243 00825*
FAF0 CIOFF  00244 00844*
FAE6 CION   00242 00835*
FAE5 CIRTN  00828 00830*
FE8F CLOAD  00518 01516*
FE9B CLVDFT 01521 01524*
FE92 CLVOFS 01516 01519*01530
F8F7 CMD    00354 00380*00439
F935 CMD2   00415*00425
F948 CMD3   00422 00424*
F95C CMDBAD 00435*00464 01288 01499 01631
F977 CMDCMP 00450*00455
F901 CMDDDL 00387*00391
F96C CMDFLS 00444*00453
F94D CMDGOT 00416 00427*
F990 CMDMEM 00420 00463*
F8F9 CMDNEP 00383*00800
F90A CMDNOL 00384 00388 00392*00462
F953 CMDSCH 00430*00434 00445
F96F CMDSIZ 00443 00446*
F967 CMDSME 00431 00441*
F99B CMDTB2 00254 00496*
F99C CMDTBL 00233 00500*
F987 CMDXQT 00410 00413 00459*00467
FDC3 CMEM   00521 01390*
FDC8 CMEM2  01392*01424 01441
FDD1 CMEM4  01397*01404 01408
FDC6 CMEMN  00465 01391*
FDE0 CMENUM 01398 01405*
FDEC CMESTR 01412*01417
FE02 CMNOTB 01420 01426*
```

```
FDE8 CMNOTC 01401 01410*
FE0E CMNOTL 01427 01434*
FDF8 CMNOTQ 01411 01419*
FE1C CMNOTU 01435 01443*
FE18 CMPADP 00411 00465 01432 01440*
FE16 CMPADS 01438*01444
FDFE CMSPCE 01414 01422*
FEB7 CNULLS 00524 01546*
FD74 CNVGOT 01325 01331*
FD62 CNVHEX 00967 01302 01322*
FD76 CNVOK  01312 01332*
FD78 CNVRTS 01287 01303 01323 01327 01329 01333*01372
FAF1 CODTA  00246 00852*
FB0F CODTAD 00869*00872
FB12 CODTAO 00854 00864 00870*
FB07 CODTLP 00864*00866
FB03 CODTPD 00859 00861*
FB0D CODTRT 00856 00867*
FEC8 COFFS  00527 01561*
FEDF COFNO1 01572 01575*
FF8E CONV1  01645 01683*
FFB7 CONV2  01662 01691*
FAF0 COOFF  00247 00845*
FAE6 COON   00245 00836*
FE71 CPUNCH 00530 01502*
000D CR     00038*00427 00621 00667 00858 01034 01166 01185 01428 01496 01654
FC4A CREG   00533 01102*
FEBC CSTLEV 00536 01551*
FEA4 CTRACE 00539 01535*
FEAA CTRCE3 00766 01538*
FEA1 CVER   00542 01530*
FE3E CWINDO 00545 01469*
DF8E DELIM  00153*00751 00757 01223 01236 01256
0000 DFTCHP 00026*00257
0005 DFTNLP 00027*00257
0010 DLE    00039*00855
0004 EOT    00035*00343 00652 00684 00738 00782 01032 01034
FABD ERRMSG 00436 00782*00789
FACE ERROR  00314 00789*
FCE9 EXP1   00253 01232*
FD07 EXP2   01234 01250*01251
FD23 EXPADD 01266*01282
FD17 EXPCDL 01252 01260*01269
FD2B EXPCHM 01262 01270*
FCEB EXPDLM 01233*01237
FD05 EXPRTN 01248*01257 01275
FD36 EXPSUB 01271 01276*
FD0D EXPTDI 01254*01273
FD0F EXPTDL 01241 01244 01247 01255*
FD42 EXPTRM 01263 01276 01286*
FFE0 FIRQ   01706*01720
FABC FIRQR  00237 00816*
FD83 GOADDR 01344 01349*01380
FDA2 GONDFT 01351 01367*
0034 HIVTR  00100*00592
FC00 HSBLNK 01046*01049
FC47 HSDRTN 01062 01086 01092*
FBFC HSDTA  00248 01043*01091
```

```
FC28 HSHCHR 01076*01084
FC35 HSHCOK 01079 01081*
FC33 HSHDOT 01077 01080*
FC14 HSHLNE 01060*01090
FC20 HSHNXT 01068*01071
FC06 HSHTTL 01051*01059
0000 INCHNP 00056*00920 00924 00966 01337 01647 01653
F844 INITVT 00188 00233*
F87D INTVE  00197 00264*
F870 INTVS  00197 00256*
FFE4 IRQ    01707*01721
FAD8 IRQR   00238 00808*
DF99 LASTOP 00139*00752 01539
FAC1 LDDP   00297 00740 00784*00809
000A LF     00037*00623 00638 00669 01034 01426
DF8F MISFLG 00151*00402 00619 00741 00772 00886 00897 01364
0008 MONITR 00064*00222
FA79 MSHOWP 00738*00748
FE36 MUPBAD 01459 01462*
FE2B MUPDAT 01406 01416 01456*
FFEC NMI    01709*01723
FAB7 NMICON 00742 00772*
FA7D NMIR   00240 00740*
FAB0 NMITRC 00744 00747 00766*
DF9B NUMBER 00137*00401 00466 01179 01255 01260 01265 01267 01278 01299 01300
            01308 01309 01405 01497 01637
0008 NUMBKP 00029*00126 00128 00389 01358 01374 01620 01633
000B NUMFUN 00068*00313
001B NUMVTR 00099*00124 00190
0004 OUT2HS 00060*01069 01156 01574 01677
0005 OUT4HS 00061*00754 01065 01153 01452 01579 01612
0001 OUTCH  00057*00396 00885 00893 00896 00983 01082 01142 01146 01396 01430
            01465
000B PAUSE  00067*
DFFC PAUSER 00117*00252
DF93 PCNTER 00145*00393 01242
0006 PCRLF  00062*00381 01044 01061 01093 01161 01439 01581 01616 01679
0003 PDATA  00059*00352 00791 00999 01023
0002 PDATA1 00058*00438 00750
003E PROMPT 00028*00394
FE21 PRTADR 01440 01448*
DF95 PSTACK 00143*00398 00435
E000 PTM    00025*00042 00043 00044 00045 00046 00047 00259 00355 00356 00358
            00359 00361 01542
E000 PTMC13 00043*00359
E001 PTMC2  00044*00358 00361
E001 PTMSTA 00042*
E002 PTMTM1 00045*00355 00356 01542
E004 PTMTM2 00046*
E006 PTMTM3 00047*
E700 RAMOFS 00021*00111
FD79 READ   00407 00424 01258 01301 01336*01412
FC94 REG4   01157*01176 01186
FCC3 REGAGN 01167 01189*
FC70 REGCHG 01104 01135*
FC9D REGCNG 01149 01164*
FC50 REGMSK 01123*01137
FCB1 REGNXC 01165 01177*
```

```
FC78 REGP1   01138*01143 01159
FC81 REGP2   01140 01144*
FC92 REGP3   01151 01155*
FAB3 REGPRS  00755 00768*00799
FC6F REGPRT  00768 01102 01134*
FC9B REGRTN  01162*01201
FCAA REGSKP  01172*01175
FCC9 REGTP1  01191*01194
FCD6 REGTP2  01197*01200
FCBB REGTWO  01181 01183*
F837 RESET   00217*00241 01724 01726
F83D RESET2  00219*00223
F000 ROM2OF  00023*00202
DF66 ROM2WK  00155*
F800 ROMBEG  00020*00023 00111 00167 01716
0800 ROMSIZ  00022*00023 01716
FFD4 RSRVD   01703*01717
FAD8 RSRVDR  00234 00809*
DF97 RSTACK  00141*00345 00788
FABC RTI     00774*00816
FAF0 RTS     00787 00841*00844 00845
F9EC SEND    00568*00624 00640 00668 00682
F8C9 SIGNON  00342*00350
008C SKIP2   00049*00863 01154 01220 01523
DFF8 SLEVEL  00123*00746 01553 01556
0007 SPACE   00063*01047 01054 01056 01073 01173 01423
DF51 STACK   00158*00217
FEC3 STLDFT  01551 01555*
FFE8 SWI     01708*01722
FFDC SWI2    01705*01719
FAD8 SWI2R   00236 00806*
FFD8 SWI3    01704*01718
FAD8 SWI3R   00235 00807*
DFFB SWIBFL  00119*00301 00311 01363
DF90 SWICNT  00149*00296 00641 00743
F8B5 SWIDNE  00302 00306 00311*
F8A8 SWILP   00305*00308
F895 SWIR    00239 00296*
F87D SWIVTB  00283*00283 00284 00285 00286 00287 00288 00289 00290 00291 00292
             00293 00294 00317
DF91 TRACEC  00147*00403 00759 00762 01536
DF51 TSTACK  00157*01189
0009 VCTRSW  00065*
DFC2 VECTAB  00125*00183 00348 00349 00353 00429 00432 00568 00594 00625 00724
             00725 00825 00837 00853 00857 00860 00977 00981 00985 01025 01224
             01485 01507 01508 01510 01540 01547 01703 01704 01705 01706 01707
             01708 01709
DFA0 WINDOW  00131*01245 01470
DF00 WORKPG  00111*00112 00113
FA72 XQCIDT  00612 00709 00716 00725*
FA6E XQPAUS  00611 00700 00715 00724*00869
FAD5 ZBKCMD  00756 00758 00760 00763 00765 00800*
FAD3 ZBKPNT  00293 00310 00799*00810
FA2A ZIN2    00622 00625*
FA11 ZINCH   00283 00612*00615 00617
FA0F ZINCHP  00611*00613
F8E6 ZMONT2  00347 00353*
F8D2 ZMONTR  00291 00345*
```

```
F9F2 ZOT2HS 00287 00571*
F9F0 ZOT4HS 00288 00570*
FA2E ZOTCH1 00284 00636*
FA37 ZOTCH2 00582 00640*
FA39 ZOTCH3 00593 00598 00600 00620 00626 00641*00704
F9D9 ZOUT2H 00557*00570 00571 01030 01393
F9E6 ZOUTHX 00561 00564*01052
FA4E ZPAUSE 00294 00700*
FA3D ZPCRLF 00289 00654*
FA3C ZPCRLS 00637 00652*00654
FA40 ZPDATA 00286 00667*
FA48 ZPDTA1 00285 00683*
FA46 ZPDTLP 00639 00682*00685
F9F6 ZSPACE 00290 00581*
F9FA ZVSWTH 00292 00591*
```

附录14 6809单板机监控程序J-MONITOR①

监控程序是使微型计算机特别是单板微型机工作的一种小型操作系统。这对6809系统的应用有重要意义。这里给出的监控程序J-MONITOR的清单可使用户对监控程序有进一步的了解，需要时可以增加命令，以方便自己的使用。本监控程序命令有以下10种：

（1）断点的设置和删除

（2）装入

（3）存储器内容显示/修改

（4）寄存器内容显示

（5）存储器内容显示

（6）执行

（7）单步执行

（8）跟踪

（9）寄存器内容修改

(10) 复制（拷具）

1. **命令简要用法**

（1）断点的设置和删除（Break）

> B<adr 1>, <adr 2>, …<adr 8> CR

断点可以设置到8个。

例：

>B 2004 CR（CR表示回车键）

>B 2006 CR

>B 2008 CR

>B 200A CR

① 见インターフエース杂志 1983年第10期 P.272～280

>B 200B CR

>B 2011 CR

>B 2012 CR

>B 2013 CR

以上断点的设置要一个个进行，为了再确认一遍所设置的断点，可以使用间隔（空格）键进行。

>B SP (SP表示间隔键)

这时即可显示出内容是：

>B 2004 2006 2008 200A 200B 2011 2012 2013

需要删除断点时，可以输入回车键，即

>B CR

（2）装入（Load）

本程序是通过ACIA使程序往存储器进行装入的。其命令是：

>L <偏值> CR

所装入的程序如果需要偏值，可以给出偏值的大小。

（3）存储器内容显示/修改（Memory Change）

>M <adr 1>, <adr 2> CR

其中adr 1表示存储器的起始地址，adr 2表示存储器的终止地址。

例：

>M 1000, 1010 CR

1000 22

1001 84

⋮

1010 10

若需修改内容时，可以进行以下操作：

>M 1000 CR

1000 22 A6

其中22为当前值，A6为修改值。

1001 84 CR

1002 11 SP

1001 84

以上内容说明如果在显示地址内容时，按间隔键SP时，可以使地址往回走一个单元。

（4）寄存器内容显示（Register Display）

>R CR

P—F000 X—0000 Y—0000 A—01 B—00 C—D0 DP—00 U—FF6A S—FF8B

（5）存储器内容显示（Dump）

>D <adr 1>, <adr 2>, …<adr 8> CR

该命令的作用是在跟踪或断点操作时，除去寄存器内容外，还想知道那些有关地址的内容而使用的命令。该命令最多可设置8个点。

例：

>D 2008 CR

>D 2009 CR

⋮

>D 200A CR

以上对设置的确认和全部内容的删除方法和断点命令的操作相同。

（6）执行 (Execution)

设有两种操作形式：

A. >E <起始adr.> CR

B. >E CR

第一种形式是给出程序要执行的起始地址；第二种形式，在作断点操作过程中，要暂时停止再作断点操作，而要进行继续执行时，可以使用这种方式。

（7）单步执行 (Single Step)

在进行程序调试过程中，程序计数器每次增加，那时需要一边确认各寄存器的内容，一边还要继续执行，该命令就是为这种情况而设置的。该命令通常与跟踪命令配合使用，每次进行一步。

（8）跟踪 (Trace)

>T <起始adr>, <16位数> CR

使用该命令时，只要分别给定起始地址和16位的数值即可进行，可以显示所有寄存器的内容。

（9）寄存器内容修改 (Register)

这是给寄存器可以设置任意数值的命令。

>, A-50 11 CR（给A寄存器输入11）

>R CR

>P—F000 X—0000 A—11 B—00 C—50 DP—00 U—0000 S—FFB7

>, B—10 55 CR（给B寄存器输入55）

（10）复制（拷贝） (Copy)

>C 1000,1100,1500

该命令的含意是把1000～1100地址中的内容复制到（又称存储器内容搬家）从1500地址开始的存储器区之中。

该命令通常在程序调试过程中，要把命令插入到程序之中进行使用。也就是说，首先要用复制命令把程序移走，在此之后再用存储器修改命令，插入所希望的命令。而且，该命令在使处于ROM中的程序往RAM中移入时也需要使用。

2. 6809 J-MONITOR程序清单

附表14.1 6809调试／监控程序清单

```
00001                         *
00002                         *
00003                         **********************************
00004                                OPT     LLEN=120
00005                         *
00006                         *    M6809 MONITOR
00007                         *
00008                         *
00009                         **********************************
00010                         *
00011              FCF4     A ACIASC EQU     $FCF4     ; ACIA STATUS ( DEBUG=$A600,DOS=$FCF4 )
00012              FCF5     A ACIADT EQU     ACIASC+1  ; SDATA
00013                         *
00014              FCC0     A PTM    EQU     $FCC0     ; PROGRAMMABLE TIMMER MODULE
00015                         *             ;  (DEBUG=$A500,DOS=FCC0 )
00016                         *
00017              003F     A SWI    EQU     $3F       ; SWI INSTRUCTION
00018                         *
00019                         * ASCII CHARACTER
00020                         *
00021              0017     A CNTW   EQU     $17
00022              0004     A EOT    EQU     4
00023              000D     A CR     EQU     $D
00024              000A     A LF     EQU     $A
00025              0020     A SP     EQU     $20
00026              0018     A CNTX   EQU     $18
00027              0008     A RUBOUT EQU     $08
00028                         *
00029                         * MACRO DEFINITION
00030                         *
00031                         SKIP2  MACR
00032                          FCB $8C ; CMPX
00033                          ENDM
00034                         *
00035                         DBNE   MACR
00036                          DECB
00037                          BNE \0
00038                          ENDM
00039                         *
00040                         *
00041                         *
00042A FF00                          ORG     $FF00     :
00043                         *
00044A FF00        0002     A BEGADR RMB     2         ; BEGIN ADR
00045A FF02        0002     A ENDADR RMB     2         ; END ADR
00046A FF04        0002     A SCRACH RMB     2         ; TEMP AREA
00047A FF06        0001     A CHKSUM RMB     1         ; CHECKSUM BUFFER
00048                         *
00049A FF07        0001     A TRFLAG RMB     1         ; TRACE FLAG
00050A FF08        0001     A ABFLAG RMB     1         ; ABEND FLAG (M= ".P= " )
00051                         *                        :
00052A FF09        0002     A PACKET RMB     2
00053A FF0B        0002     A        RMB     2         ;
00054A FF0D        0002     A BUFFR  RMB     2
00055                         *
00056              0000     A POFST  EQU     0
00057A FF16                          ORG     $FF16
00058                         *
00059A FF16        0002     A .PSAVE RMB     2         ; PC
00060              0002     A UOFST  EQU     *-.PSAVE
00061A FF18        0002     A .USAVE RMB     2         ; IU
00062              0004     A YOFST  EQU     *-.PSAVE
```

```
00063A FF1A       0002    A .YSAVE RMB    2         ; IY
00064             0006    A XOFST  EQU    *-.PSAVE
00065A FF1C       0002    A .XSAVE RMB    2         ; IX
00066             0008    A DOFST  EQU    *-.PSAVE
00067A FF1E       0001    A .DSAVE RMB    1         ; DP
00068             0009    A AOFST  EQU    *-.PSAVE
00069A FF1F       0001    A .ASAVE RMB    1         ; A REG
00070             000A    A BOFST  EQU    *-.PSAVE
00071A FF20       0001    A .BSAVE RMB    1         ; B REG
00072             000B    A COFST  EQU    *-.PSAVE
00073A FF21       0001    A .CSAVE RMB    1         ; CCR
00074             000C    A SOFST  EQU    *-.PSAVE
00075A FF22       0002    A .SSAVE RMB    2         ; SP
00076                       *
00077A FF24       0001    A DMPCNT RMB    1         ; DUMP COUNT
00078A FF25       0010    A DMPTBL RMB    16        ; DUMP ADDRESS TABLE
00079A FF35       0001    A BRKCNT RMB    1         ; BREAK POINT COUNT
00080A FF36       0010    A BRLTBL RMB    16        ; BREAK POINT ADDRESS
00081A FF46       0010    A        RMB    16        ; YSER INSTRUCTION
00082A FF56       0010    A BRKPCN RMB    16        ; BREAK PASS COUNT
00083A FF66       0001    A COMND  RMB    1         ; COMMAND BUFFER
00084A FF67       0001    A LSTCHR RMB    1         ; INPUT LAST CHARACTER
00085A FF68       0001    A PCOUNT RMB    1         ; OPERAND COUNT
00086A FF69       0001    A COUNT  RMB    1         ; CHARACTER COUNT
00087A FF6A       0002    A OPRND1 RMB    2         ; OPR1
00088A FF6C       0002    A OPRND2 RMB    2         ; OPR2
00089A FF6E       0002    A OPRND3 RMB    2         ; OPR3
00090A FF70       0006    A        RMB    6         ; NEX OPR
00091             FF76    A TBLEND EQU    *         ;
00092A FF8F                        ORG    $FF8F    EXBUG STACK
00093             FF8F    A STACK  EQU    *
00094A FF8F       0028    A        RMB    40
00095             FFB7    A USTACK EQU    *
00096                       *
00098                       *
00099                       * ENTRY POINTS
00100                       *
00101                       *
00102A F000                        ORG    $F000
00103                       *
00104A F000 7E    F02D    A POWUP  JMP    START     ; RESET
00105A F003 7E    F000    A XBEGIN JMP    POWUP
00106A F006 7E    F6B7    A XCBCDH JMP    ASBIN
00107A F009 7E    F682    A XCHEXL JMP    CHEXL     ;
00108A F00C 7E    F68C    A XCHEXR JMP    CHEXR
00109A F00F 7E    F000    A XINADD JMP    POWUP
00110A F012 7E    F722    A XINCH  JMP    CI        ; CONSOL IN NON WITH PARITY
00111A F015 7E    F6DF    A XINCHN JMP    RDCHR     ; CONSOL IN NON PARITY
00112A F018 7E    F6CD    A XOUTCH JMP    WRCHR     ; CONSOL OUT
00113A F01B 7E    F67C    A XOUT2H JMP    OUT2HS    ; PRINT 2 HEX CHARC
00114A F01E 7E    F67A    A XOUT4H JMP    OUT4HS    ; PRINT 4 HEX
00115A F021 7E    F6C4    A XPCRLF JMP    CRLF
00116A F024 7E    F66D    A XPDATA JMP    STRNG     ;PRINT DATA STRING
00117A F027 7E    F670    A XDATI  JMP    STRNG2    ; PRINT DATA STRING
00118A F02A 7E    F6CB    A XPSPAC JMP    SPACE     ; PRINT SPACE
00119                       *
00121                       *
00122                       *  PRIGRAM START
00123                       *
00124A F02D 10CE FFB7     A START  LDS    #USTACK
00125                       *
00126                       * INITIALIZE PIA,PTM
00127                       *
00128A F031 86    01      A        LDA    #1        ; SELECT CR1 ; PTM INITIAL
00129A F033 B7    FCC1    A        STA    PTM+1
00130A F036 86    A3      A        LDA    #%10100011 ; PRESET
00131A F038 B7    FCC0    A        STA    PTM
00132A F03B CE    FCF4    A        LDU    #ACIASC   ; ACIA ADDRESS
00133A F03E C6    03      A        LDB    #$3      RESET ACIA
00134A F040 E7    C4      A        STB    0,U
00135A F042 C6    51      A        LDB    #$51
00136A F044 E7    C4      A        STB    0,U      ACIASC
```

```
00137A F046 17    071E F767 IACIA1 LBSR   TBLCLR    ; TABLE CLEAR
00138A F049 7F    FF1E    A        CLR    .DSAVE    SET PSEUDO DPR=0
00139A F04C 86    50      A        LDA    #$50      SET PSEUDO I, F MASK
00140A F04E B7    FF21    A        STA    .CSAVE
00141A F051 10CE  FF8F    A RENTER LDS    #STACK    ; INIT SP
00142A F055 8E    FFB7    A        LDX    #USTACK   INIT PSEUDO SP
00143A F058 BF    FF22    A        STX    .SSAVE
00144A F05B CE    0000    A RENTR1 LDU    #0        U=TOP OF EXBUG'S POINTER
00145                       *
00146A F05E 8E    F4FF    A ENTER  LDX    #NMISRV   INIT NMI VECTOR
00147A F061 BF    FFFC    A        STX    $FFFC
00148A F064 8E    F3E4    A        LDX    #SWISRV   INIT SWI VECTOR
00149A F067 BF    FFFA    A        STX    $FFFA
00150A F06A 8E    F7DA    A        LDX    #HEAD     ; PRINT HEADING
00151A F06D 8D    B5   F024        BSR    XPDATA
00152A F06F 10CE  FF8F    A COMAND LDS    #STACK    ; INIT SP
00153A F073 8E    F7A8    A        LDX    #PROMPT
00154A F076 8D    AC   F024        BSR    XPDATA
00155A F078 BD    F6C0    A        JSR    ECHO      ; GET COMMAND
00156A F07B 81    2E      A        CMPA   #'.       ; RENTER?
00157A F07D 27    F0   F06F        BEQ    COMAND    ; RENTER
00158A F07F 81    2C      A        CMPA   #'.       ; REG COMMAND
00159A F081 1027  02DE F363        LBEQ   DOTCMD    ;
00160A F085 17    06BB F743        LBSR   HEXCHK    ; ALFCHK
00161A F088 25    E5   F06F        BCS    COMAND    ;
00162A F08A 2A    E3   F06F        BPL    COMAND    ; DIGIT
00163A F08C B7    FF66    A        STA    COMND     ; SAVE
00164A F08F BD    F6CB    A        JSR    SPACE     ;
00165A F092 17    0557 F5EC        LBSR   SETOPR    ; OPERAND
00166A F095 8E    F772    A        LDX    #CMDTBL   ; COMMAND TABLE
00167A F098 B6    FF66    A        LDA    COMND     ; LOAD
00168A F09B A1    84      A COMND1 CMPA   ,X        ;
00169A F09D 27    0A   F0A9        BEQ    COMND2    ;
00170A F09F 30    03      A        LEAX   3,X       ; NEXT-SERCH
00171A F0A1 8C    F7A8    A        CMPX   #CMDEND   ; TABLE END
00172A F0A4 26    F5   F09B        BNE    COMND1    ; CONTINUE
00173A F0A6 7E    F658    A        JMP    ERR1      ; EXIT
00174A F0A9 6E    98 01   A COMND2 JMP    [,X]      ; JUMP
00175                       *
00176                       *
00177                       *
00178             F0AC    A ACMND  EQU    *
00179                       *
00180                       ***********************************
00181                       *
00182                       * B-REAK POINT SET COMMAND
00183                       *
00184                       * <B> <ADR1>,<ADR2>,----<ADR8>
00185                       *
00186             F0AC    A BCMND  EQU    *
00187A F0AC CE    FF35    A        LDU    #BRKCNT   ; BREAK COUNT,TABLE
00188A F0AF 20    30   F0E1        BRA    DCMD02    ; CHAIN-
00189                       *
00190                       ******************************
00191                       *
00192                       * COPY COMMAND
00193                       * <C> <START ADR>,<END ADR>,<DATA'BYTE'>
00194                       *
00195             F0B1    A CCMND  EQU    *
00196A F0B1 F6    FF68    A        LDB    PCOUNT
00197A F0B4 C1    03      A        CMPB   #3
00198A F0B6 1026  059E F658        LBNE   ERR1      ; SYNTAX ERR
00199A F0BA BE    FF6A    A        LDX    OPRND1
00200A F0BD 10BE  FF6C    A        LDY    OPRND2
00201A F0C1 BC    FF6C    A        CMPX   OPRND2
00202A F0C4 25    02   F0C8        BCS    CCMD02    ; OPRND1<OPRND2
00203A F0C6 1E    12      A        EXG    X,Y       ; OPRND1>=OPRND2
00204A F0C8 BF    FF00    A CCMD02 STX    BEGADR
00205A F0CB 10BF  FF02    A        STY    ENDADR
00206A F0CF FE    FF6E    A        LDU    OPRND3    ; TO COPY ADDRESS
00207A F0D2 A6    80      A CCMD04 LDA    ,X+
```

```
00208A F0D4 A7    C0        A          STA    ,U+
00209A F0D6 BC    FF02      A          CMPX   ENDADR
00210A F0D9 23    F7   F0D2            BLS    CCMD04
00211A F0DB 7E    F06F      A          JMP    COMAND
00212                             *
00213                             ********************************
00214                             *
00215                             * D-UMP COMMAND
00216                             * <D> <ADR1>,<ADR2>,----<ADR8>
00217                             *
00218             F0DE      A DCMND    EQU    *
00219A F0DE CE    FF24      A DCMD02   LDU    #DMPCNT    ; DUMP COUNT,TABLE
00220A F0E1 B6    FF67      A DCMD02   LDA    LSTCHR     ; LAST CHARACTER
00221A F0E4 81    0D        A          CMPA   #CR
00222A F0E6 26    3D   F125            BNE    DCMD14     ;
00223A F0E8 F6    FF68      A          LDB    PCOUNT     : PARAMETER
00224A F0EB 27    2F   F11C            BEQ    DCMD10     ; TABLE CLEAR
00225A F0ED 8E    FF6A      A          LDX    #OPRND1    ; NOT
00226A F0F0 20    04   F0F6            BRA    DCMD04
00227                             *
00228                             * EXBUG MAID ENTRY POINT
00229                             *
00230A F0F3                                ORG    $F0F3
00231                             *
00232A F0F3 7E    F06F      A          JMP    COMAND
00233                             *
00234A F0F6 34    40        A DCMD04   PSHS   U          ; SAVE
00235A F0F8 10AE  81        A          LDY    ,X++       ; BRK ADR
00236A F0FB E6    C0        A          LDB    ,U+        ; TABLE
00237A F0FD 27    0E   F10D            BEQ    DCMD07     ; EMPTY
00238A F0FF C1    08        A          CMPB   #8         ; TABLE OVER?
00239A F101 1024  0553 F658            LBCC   ERR1       ; YES
00240A F105 10AC  C1        A DCMD06   CMPY   ,U++
00241A F108 27    08   F112            BEQ    DCMD08     ; SAME(DOUBLE)
00242A F10A                            DBNE   DCMD06
00243A F10D 10AF  C4        A DCMD07   STY    ,U         ; NEW
00244A F110 6C    F4        A          INC    [,S]       ; COUNT UP
00245A F112 35    40        A DCMD08   PULS   U          ; RESTORE
00246A F114 7A    FF68      A          DEC    PCOUNT     ; END?
00247A F117 26    DD   F0F6            BNE    DCMD04     ; NOT
00248A F119 7E    F06F      A DCMD09   JMP    COMAND     ; EXIT
00249A F11C C6    11        A DCMD10   LDB    #16+1      ; 8 POINT
00250A F11E 6F    C0        A DCMD11   CLR    ,U+
00251A F120                            DBNE   DCMD11
00252A F123 20    F4   F119            BRA    DCMD09     ; EXIT
00253A F125 81    20        A DCMD14   CMPA   #SP        ; DISPLAY?
00254A F127 1026  052D F658            LBNE   ERR1       ; NOT
00255A F12B 33    41        A          LEAU   1,U        ; TABLE
00256A F12D C6    08        A          LDB    #8         ; POINT COUNT
00257A F12F 1F    31        A          TFR    U,X
00258A F131 17    0546 F67A DCMD16     LBSR   OUT4HS     ; OUT
00259A F134                            DBNE   DCMD16
00260A F137 20    E0   F119            BRA    DCMD09
00261                             *
00262                             *
00263                             *
00264                             ********************************
00265                             *
00266                             * E XCUTION COMMAND
00267                             * <E> <START ADR> -- JUMP
00268                             * <E>             -- CONTINUE
00269                             *
00270             F139      A ECMND    EQU    *
00271A F139 F6    FF68      A          LDB    PCOUNT     ; OPERAND COUNT
00272A F13C 1027  0319 F459            LBEQ   SWIS16     ; CONTINUE
00273A F140 C1    01        A          CMPB   #1
00274A F142 1026  0512 F658            LBNE   ERR1       ; SYNTAX ERR
00275A F146 BE    FF6A      A ECMD01   LDX    OPRND1     ; START ADDRESS
00276A F149 BF    FF16      A          STX    .PSAVE
00277A F14C 17    0321 F470            LBSR   SWAP       ; SWI INSTRUCTION SET
00278A F14F 10FE  FF22      A ECMD04   LDS    .SSAVE     ; SP
```

```
00279A F153 BE   FF16     A          LDX   .PSAVE      ; PC
00280A F156 34   10       A          PSHS  X           ; PUSH
00281A F158 B6   FF1E     A          LDA   .DSAVE      ; DP
00282A F15B 34   02       A          PSHS  A           ; PUSH
00283A F15D B6   FF21     A          LDA   .CSAVE      ; CC
00284A F160 34   02       A          PSHS  A           ; PUSH
00285A F162 BE   FF1C     A          LDX   .XSAVE      ; IX
00286A F165 10BE FF1A     A          LDY   .YSAVE      ; IY
00287A F169 FE   FF18     A          LDU   .USAVE      ; IU
00288A F16C 35   89       A          PULS  CC,DP,PC    ; JUMP
00289                       *
00290                       **********************************
00291                       *
00292            F16E     A ICMND    EQU   *
00293A F16E F6   FF68     A          LDB   PCOUNT      ;
00294A F171 C1   03       A          CMPB  #3          ;
00295A F173 1026 04E1 F658           LBNE  ERR1        ; SYNTAX ERR
00296A F177 FC   FF6E     A          LDD   OPRND3
00297A F17A 4D                       TSTA
00298A F17B 1026 04D9 F658           LBNE  ERR1        ; DATA=WORD
00299A F17F BE   FF6A     A          LDX   OPRND1
00300A F182 10BE FF6C     A          LDY   OPRND2
00301A F186 BC   FF6C     A          CMPX  OPRND2
00302A F189 25   02   F18D           BCS   ICMD04      ; OPRND1<OPRND2
00303A F18B 1E   12       A          EXG   X,Y         ; OPRND1>=OPRND2
00304A F18D BF   FF00     A ICMD04   STX   BEGADR
00305A F190 10BF FF02     A          STY   ENDADR
00306A F194 E7   80       A ICMD06   STB   ,X+
00307A F196 BC   FF02     A          CMPX  ENDADR
00308A F199 23   F9   F194           BLS   ICMD06
00309A F19B 7E   F06F     A          JMP   COMAND
00310                       *
00311                       *
00312                       *
00313                       **********************************
00314                       *
00315                       * L-OAD COMMAND
00316                       *
00317                       *  <L><OFSET>
00318                       *
00319            F19E     A LCMND    EQU   *
00320A F19E B6   FF67     A          LDA   LSTCHR      ; LAST CHARACTER
00321A F1A1 81   0D       A          CMPA  #CR         ; DELLIMITOR?
00322A F1A3 1026 04B1 F658           LBNE  ERR1        ; NOT
00323A F1A7 7F   FF00     A          CLR   BEGADR
00324A F1AA 7F   FF01     A          CLR   BEGADR+1    ; INITIAL
00325A F1AD B6   FF68     A          LDA   PCOUNT      ; OFSET EXIST?
00326A F1B0 27   0C   F1BE           BEQ   LCMD01      ; NOT
00327A F1B2 81   01       A          CMPA  #1          ; YES-OK
00328A F1B4 1026 04A0 F658           LBNE  ERR1        ; NOT
00329A F1B8 BE   FF6A     A          LDX   OPRND1      ; YES
00330A F1BB BF   FF00     A          STX   BEGADR      ; SAVE
00331A F1BE 86   11       A LCMD01   LDA   #$11
00332A F1C0 B7   FCF4     A          STA   ACIASC      ; CHANGE
00333A F1C3 7F   FF06     A          CLR   CHKSUM      ; INITIAL
00334A F1C6 17   0516 F6DF           LBSR  RDCHR       ;
00335A F1C9 81   53       A          CMPA  #'S         ; START OF RECORD?
00336A F1CB 26   F1   F1BE           BNE   LCMD01      ; NOT-WAIT
00337A F1CD 17   050F F6DF LCMD02    LBSR  RDCHR       ; TYPE
00338A F1D0 81   39       A          CMPA  #'9         ; END OF FILE?
00339A F1D2 27   56   F22A           BEQ   LCMD14      ; YES
00340A F1D4 81   31       A          CMPA  #'1         ; DATA RECORD?
00341A F1D6 27   1C   F1F4           BEQ   LCMD06      ; YES
00342A F1D8 81   30       A          CMPA  #'0         ; START OF HEADING?
00343A F1DA 26   F1   F1CD           BNE   LCMD02      ; NOT-WAIT
00344A F1DC 17   04C1 F6A0           LBSR  BYTE        ; COUNT- DUMMY
00345A F1DF C6   04       A          LDB   #4
00346A F1E1 F7   FF69     A          STB   COUNT       ;
00347A F1E4 17   04AF F696           LBSR  WORD        ; DUMMY
00348A F1E7 17   04B6 F6A0 LCMD04    LBSR  BYTE
00349A F1EA 17   04E0 F6CD           LBSR  WRCHR       ; ECHO
```

```
00350A F1ED 7A    FF69    A         DEC    COUNT
00351A F1F0 26    F5    F1E7        BNE    LCMD04    ; CONTINUE
00352A F1F2 20    CA    F1BE        BRA    LCMD01    ;
00353A F1F4 17    04A9 F6A0 LCMD06 LBSR   BYTE      ; DATA COUNT
00354A F1F7 4A                      DECA
00355A F1F8 B7    FF69    A         STA    COUNT
00356A F1FB 17    0498 F696         LBSR   WORD      ; ADDRESS
00357A F1FE F3    FF00    A         ADDD   BEGADR
00358A F201 7A    FF69    A         DEC    COUNT
00359A F204 7A    FF69    A         DEC    COUNT     ; FOR 2 BYTES
00360A F207 1F    01      A         TFR    D,X       ;
00361A F209 17    0494 F6A0 LCMD08 LBSR   BYTE
00362A F20C A7    84      A         STA    ,X
00363A F20E A1    80      A         CMPA   ,X+       ; VERIFY
00364A F210 26    14    F226        BNE    LCMD12    ; MESSAGE
00365A F212 7A    FF69    A         DEC    COUNT
00366A F215 26    F2    F209        BNE    LCMD08    ;
00367A F217 17    0486 F6A0         LBSR   BYTE
00368A F21A B6    FF06    A         LDA    CHKSUM
00369A F21D 4C                      INCA
00370A F21E 26    02    F222        BNE    LCMD10    ; CHECKSUM ERR
00371A F220 20    9C    F1BE        BRA    LCMD01    ; CONTINUE
00372A F222 8E    F7B5    A LCMD10 LDX    #CKSUM
00373A F225       10      A         FCB    $10
00374A F226 8E    F7C4    A LCMD12 LDX    #VERFY
00375A F229       10      A         FCB    $10
00376A F22A 8E    F7D1    A LCMD14 LDX    #LDEND
00377A F22D 86    51      A         LDA    #$51
00378A F22F B7    FCF4    A         STA    ACIASC
00379A F232 17    0438 F66D         LBSR   STRNG
00380A F235 7E    F06F    A         JMP    COMAND
00381                       *
00382                       *
00383                       *********************************
00384                       *
00385                       * M-EMORY COMMAND
00386                       * <M> <ADR1>,<ADR2>
00387                       *
00388             F238    A MCMND  EQU    *
00389A F238 BE    FF6A    A         LDX    OPRND1
00390A F23B BF    FF00    A         STX    BEGADR    ;
00391A F23E BE    FF6C    A         LDX    OPRND2
00392A F241 BF    FF02    A         STX    ENDADR
00393A F244 F6    FF68    A         LDB    PCOUNT    ; OPERAND NUMBER
00394A F247 C1    03      A         CMPB   #3
00395A F249 1024  040B F658         LBCC   ERR1      ; OVER OPERAND
00396A F24D C1    02      A         CMPB   #2
00397A F24F 25    54    F2A5        BCS    MCMD10    ; PRINT
00398A F251 CE    FF00    A         LDU    #BEGADR   ;
00399A F254 EC    C4      A         LDD    ,U
00400A F256 C4    F0      A         ANDB   #$F0
00401A F258 ED    C4      A         STD    ,U
00402A F25A EC    42      A         LDD    2,U
00403A F25C CA    0F      A         ORB    #$0F
00404A F25E ED    42      A         STD    2,U
00405A F260 A3    C4      A         SUBD   ,U
00406A F262 1025  03F2 F658         LBCS   ERR1      ; ADDRESS ERR
00407A F266 17    047B F6E4 MCMD2  LBSR   BRKCHK    ; BREAK CHECK
00408A F269 26    37    F2A2        BNE    MCMD8     ; EXIT
00409A F26B 8E    FF00    A         LDX    #BEGADR   ; HEAD
00410A F26E 17    0409 F67A         LBSR   OUT4HS    ; PRINT
00411A F271 C6    10      A         LDB    #16
00412A F273 BE    FF00    A         LDX    BEGADR    ;
00413A F276 17    0403 F67C MCMD4  LBSR   OUT2HS    ; DATA
00414A F279                         DBNE   MCMD4
00415A F27C BE    FF00    A         LDX    BEGADR    ;
00416A F27F C6    10      A         LDB    #16
00417A F281 A6    80      A MCMD5  LDA    ,X+
00418A F283 84    7F      A         ANDA   #$7F      ; PARITY MASK
00419A F285 81    20      A         CMPA   #$20
00420A F287 2D    04    F28D        BLT    MCMD6
```

```
00421A F289 81   7F      A         CMPA   #$7F
00422A F28B 2D   02   F28F         BLT    MCMD7
00423A F28D 86   2E      A MCMD6   LDA    #'.
00424A F28F 17   043B F6CD MCMD7   LBSR   WRCHR
00425A F292                        DBNE   MCMD5
00426A F295 17   042C F6C4         LBSR   CRLF      ;
00427A F298 BF   FF00    A         STX    BEGADR
00428A F29B 27   05   F2A2         BEQ    MCMD8     ;
00429A F29D BC   FF02    A         CMPX   ENDADR
00430A F2A0 25   C4   F266         BLO    MCMD2
00431A F2A2 7E   F06F    A MCMD8   JMP    COMAND    ; EXIT
00432A F2A5 B6   FF67    A MCMD10  LDA    LSTCHR
00433A F2A8 81   0D      A         CMPA   #CR
00434A F2AA 1026 03AA F658         LBNE   ERR1      ; NOT TERMINATOR
00435A F2AE 8E   FF00    A MCMD11  LDX    #BEGADR
00436A F2B1 17   03C6 F67A         LBSR   OUT4HS    ;
00437A F2B4 BE   FF00    A         LDX    BEGADR
00438A F2B7 17   03C2 F67C         LBSR   OUT2HS    ; DATA
00439A F2BA 17   032F F5EC MCMD12  LBSR   SETOPR
00440A F2BD F6   FF68    A         LDB    PCOUNT    ; OPRAND COUNT
00441A F2C0 27   23   F2E5         BEQ    MCMD14    ; NOT
00442A F2C2 C1   02      A         CMPB   #2
00443A F2C4 1024 0390 F658         LBCC   ERR1      ; OPRAND OVER
00444A F2C8 F6   FF69    A         LDB    COUNT
00445A F2CB C1   03      A         CMPB   #3
00446A F2CD 1024 0387 F658         LBCC   ERR1      ; DATA OVER
00447A F2D1 BE   FF00    A         LDX    BEGADR    ; ADDRESS
00448A F2D4 B6   FF6B    A         LDA    OPRND1+1  ; DATA
00449A F2D7 A7   84      A         STA    ,X
00450A F2D9 A1   84      A         CMPA   ,X        ; VERIFY
00451A F2DB 27   08   F2E5         BEQ    MCMD14    ; YES
00452A F2DD 86   3F      A         LDA    #'?       ;
00453A F2DF 17   03EB F6CD         LBSR   WRCHR     ;
00454A F2E2 17   03DF F6C4         LBSR   CRLF      ;
00455A F2E5 BE   FF00    A MCMD14  LDX    BEGADR    ; GET
00456A F2E8 B6   FF67    A         LDA    LSTCHR    ; LAST CHARACTER
00457A F2EB 81   20      A         CMPA   #SP       ; BACKWORD?
00458A F2ED 27   07   F2F6         BEQ    MCMD18    ; YES
00459A F2EF 30   01      A         LEAX   1,X       ; FORWORD
00460A F2F1 BF   FF00    A MCMD16  STX    BEGADR    ;
00461A F2F4 20   B8   F2AE         BRA    MCMD11
00462A F2F6 30   1F      A MCMD18  LEAX   -1,X
00463A F2F8 BF   FF00    A         STX    BEGADR
00464A F2FB 17   03C6 F6C4         LBSR   CRLF      ; CR,LF
00465A F2FE 20   AE   F2AE         BRA    MCMD11
00466                      *
00467                      *
00468                      ***********************************
00469                      *
00470            F300    A NCMND   EQU    *
00471            F300    A PCMND   EQU    *
00472            F300    A QCMND   EQU    *
00473                      *
00474                      ***********************************
00475                      *
00476                      * R-EGISTER DISPLAY COMMAND
00477                      *
00478                      * <R><CR>
00479                      *
00480            F300    A RCMND   EQU    *
00481A F300 B6   FF67    A         LDA    LSTCHR    ;
00482A F303 81   0D      A         CMPA   #CR
00483A F305 1026 034F F658         LBNE   ERR1
00484A F309 17   0228 F534         LBSR   RDATA
00485A F30C 7E   F06F    A         JMP    COMAND
00486                      *
00487                      *
00488                      *
00489                      ***********************************
00490                      *
00491                      * S-INGLE STEP COMMAND
```

```
00492                                *
00493                                *
00494                F30F        A SCMND   EQU     *
00495A F30F B6       FF67        A         LDA     LSTCHR     ;
00496A F312 81       0D          A         CMPA    #CR
00497A F314 27       06     F31C           BEQ     SCMD02     ;
00498A F316 81       20          A         CMPA    #SP
00499A F318 1026     033C   F658           LBNE    ERR1       ; WHAT?
00500A F31C 8E       0001        A SCMD02  LDX     #1
00501A F31F BF       FF6C        A         STX     OPRND2     ; SNGLE STEP COUNT
00502A F322 F6       FF68        A         LDB     PCOUNT
00503A F325 27       2C     F353           BEQ     TCMD06     ; CONTINUE
00504A F327 C1       02          A         CMPB    #2         ;
00505A F329 1024     032B   F658           LBCC    ERR1       ; SYNTAX ERR
00506A F32D 20       1E     F34D           BRA     TCMD04
00507                                *
00508                                *
00509                                ***********************************
00510                                *
00511                                * T-RACE COMMAND
00512                                *
00513                                * <T> <START ADR>,<16BITS COUNT>
00514                                *
00515                F32F        A TCMND   EQU     *
00516A F32F B6       FF67        A         LDA     LSTCHR     ; LAST CHARACTER
00517A F332 81       0D          A         CMPA    #CR
00518A F334 1026     0320   F658           LBNE    ERR1       ; NOT
00519A F338 F6       FF68        A         LDB     PCOUNT     ; PAMETER
00520A F33B 27       16     F353           BEQ     TCMD06     ; CONTINUE
00521A F33D C1       03          A         CMPB    #3         ;
00522A F33F 1024     0315   F658           LBCC    ERR1       ; SYNTAX ERR
00523A F343 C1       01          A         CMPB    #1
00524A F345 26       06     F34D           BNE     TCMD04     ; COUNT SET
00525A F347 8E       0001        A         LDX     #1
00526A F34A BF       FF6C        A         STX     OPRND2     ; 1 STEP
00527A F34D BE       FF6A        A TCMD04  LDX     OPRND1
00528A F350 BF       FF16        A         STX     .PSAVE     ; START ADR
00529A F353 86       80          A TCMD06  LDA     #$80
00530A F355 B7       FF07        A         STA     TRFLAG     ; TRACE FLAG SET
00531A F358 B6       FF21        A         LDA     .CSAVE
00532A F35B 8A       80          A         ORA     #$80       ;
00533A F35D B7       FF21        A         STA     .CSAVE
00534A F360 7E       F42B        A         JMP     SWI508     ; CHAIN-
00535                                *
00536                F363        A UCMND   EQU     *
00537                F363        A XCMND   EQU     *
00538                F363        A YCMND   EQU     *
00539                F363        A ZCMND   EQU     *
00540                                *
00541                                *
00542                                ********************************
00543                                *
00544                                * REGISTER COMMAND
00545                                *
00546                                * <.A,B,C,D,X,Y,U,S,PC >
00547                                *
00548A F363 17       035A   F6C0 DOTCMD LBSR   ECHO       ; GET & ECHO
00549A F366 34       02          A         PSHS    A          ; SAVE
00550A F368 86       2D          A         LDA     #'-
00551A F36A 17       0360   F6CD           LBSR    WRCHR      ; WRITE
00552A F36D 35       02          A         PULS    A          ; RESTORE
00553A F36F 8E       F3C8        A         LDX     #NAMTBL    ; REGISTER NAME
00554A F372 6D       84          A DTCM02  TST     ,X
00555A F374 1027     02E0   F658           LBEQ    ERR1       ; NOT FOUND
00556A F378 A1       84          A         CMPA    ,X
00557A F37A 27       04     F380           BEQ     DTCM04     ; YEA!
00558A F37C 30       03          A         LEAX    3,X        ; NEXT
00559A F37E 20       F2     F372           BRA     DTCM02
00560A F380 EC       01          A DTCM04  LDD     1,X        ; A-OFSET,B-COUNT
00561A F382 8E       FF16        A         LDX     #.PSAVE    ; REG SAVE TABLE
00562A F385 30       86          A         LEAX    A,X
```

```
00563A F387 C1   02        A          CMPB   #2
00564A F389 34   14        A          PSHS   B,X
00565A F38B 23   05   F392            BLS    DTCM06     ; BYTE
00566A F38D 17   02EA F67A            LBSR   OUT4HS     ; WORD
00567A F390 20   03   F395            BRA    DTCM08
00568A F392 17   02E7 F67C DTCM06 LBSR   OUT2HS     ;
00569A F395 17   0254 F5EC DTCM08 LBSR   SETOPR     ; GET
00570A F398 B6   FF67      A          LDA    LSTCHR     ; LAST CHARACTER
00571A F39B 81   0D        A          CMPA   #CR
00572A F39D 1026 02B7 F658            LBNE   ERR1       ; ERR
00573A F3A1 35   14        A          PULS   B,X
00574A F3A3 B6   FF68      A          LDA    PCOUNT
00575A F3A6 27   1D   F3C5            BEQ    DTCM14     ; EXIT
00576A F3A8 81   02        A          CMPA   #2
00577A F3AA 1024 02AA F658            LBCC   ERR1       ; SYNTAX ERR
00578A F3AE F1   FF69      A          CMPB   COUNT
00579A F3B1 1025 02A3 F658            LBCS   ERR1       ; DATA OVER
00580A F3B5 C1   02        A          CMPB   #2
00581A F3B7 23   07   F3C0            BLS    DTCM12
00582A F3B9 FC   FF6A      A          LDD    OPRND1     ; WORD
00583A F3BC ED   84        A          STD    ,X
00584A F3BE 20   05   F3C5            BRA    DTCM14
00585A F3C0 FC   FF6A      A DTCM12 LDD    OPRND1
00586A F3C3 E7   84        A          STB    ,X
00587A F3C5 7E   F06F      A DTCM14 JMP    COMAND
00588                        *
00589A F3C8      41        A NAMTBL FCB    'A
00590A F3C9      0902      A          FDB    (AOFST!<8)!+2
00591A F3CB      42        A          FCB    'B
00592A F3CC      0A02      A          FDB    (BOFST!<8)!+2
00593A F3CE      43        A          FCB    'C
00594A F3CF      0B02      A          FDB    (COFST!<8)!+2
00595A F3D1      44        A          FCB    'D
00596A F3D2      0802      A          FDB    (DOFST!<8)!+2
00597A F3D4      58        A          FCB    'X
00598A F3D5      0604      A          FDB    (XOFST!<8)!+4
00599A F3D7      59        A          FCB    'Y
00600A F3D8      0404      A          FDB    (YOFST!<8)!+4
00601A F3DA      55        A          FCB    'U
00602A F3DB      0204      A          FDB    (UOFST!<8)!+4
00603A F3DD      53        A          FCB    'S
00604A F3DE      0C04      A          FDB    (SOFST!<8)!+4
00605A F3E0      50        A          FCB    'P
00606A F3E1      0004      A          FDB    (POFST!<8)!+4
00607A F3E3      00        A          FCB    0
00609                        *
00610                        *
00611                        *
00612                        *
00613                        *
00614                        *
00615                        *******************************
00616                        *
00617                        *
00618                        *********************************
00619                        *
00620                        * SWI SERVICE ROUTINE
00621                        *
00622A F3E4 1F   43        A SWISRV TFR    S,U
00623A F3E6 86   51        A          LDA    #$51
00624A F3E8 B7   FCF4      A          STA    ACIASC     ; SET
00625A F3EB 10CE FF8F      A SWIS02 LDS    #STACK
00626A F3EF 17   00B8 F4AA            LBSR   SAVE       ; USER REG SAVE
00627A F3F2 BE   FF16      A          LDX    .PSAVE
00628A F3F5 30   1F        A          LEAX   -1,X
00629A F3F7 BF   FF16      A          STX    .PSAVE     ; CURRENT
00630A F3FA 7D   FF35      A          TST    BRKCNT     ; BREAK NO
00631A F3FD 27   18   F417            BEQ    SWIS05     ; BOO!
00632A F3FF 17   0092 F494            LBSR   MATCH      ;
00633A F402 25   13   F417            BCS    SWIS05     ; BOO!
00634A F404 8E   F7A8      A          LDX    #PROMPT    ; YES!
```

```
00635A F407 20     14   F41D          BRA    SWIS06
00636A F409 7F     FF07    A SWIS04 CLR    TRFLAG     ; TRACE FLAG CLEAR
00637A F40C 8E     F7AA    A        LDX    #ABORT
00638A F40F 17     025B F66D        LBSR   STRNG      ;
00639A F412 17     02B6 F6CB        LBSR   SPACE      ; SPACE
00640A F415 20     06   F41D        BRA    SWIS06
00641A F417 8E     F7B0    A SWIS05 LDX    #SWINS
00642A F41A 17     0253 F670        LBSR   STRNG2     ; SWI-
00643A F41D 17     0114 F534 SWIS06 LBSR   RDATA      ; REG PRINT
00644A F420 7D     FF24    A        TST    DMPCNT     ; DUMP ADDRESS EXIT?
00645A F423 27     06   F42B        BEQ    SWIS08     ; NOT
00646A F425 17     029C F6C4        LBSR   CRLF       ; CR,LF
00647A F428 17     01AC F5D7        LBSR   DDATA      ; YES-OUTPUT
00648A F42B 8D     57   F484 SWIS08 BSR    RCVER      ; USER INSTRUCTION RESTORE
00649A F42D 17     02B4 F6E4        LBSR   BRKCHK     ; BREAK KEY CHECK
00650A F430 26     05   F437        BNE    SWIS10     ; BREAK
00651A F432 7D     FF07    A        TST    TRFLAG     ; NOT- TRACE FLAG
00652A F435 2B     09   F440        BMI    SWIS12     ; YES
00653A F437 7F     FF07    A SWIS10 CLR    TRFLAG     ; TRACE FLAG CLEAR
00654A F43A 7F     FF08    A        CLR    ABFLAG     ; ABEND FLAG CLEAR
00655A F43D 7E     F06F    A        JMP    COMAND     ; RENTER
00656A F440 17     0281 F6C4 SWIS12 LBSR   CRLF
00657A F443 B6     FF66    A        LDA    COMND      ;
00658A F446 81     54      A        CMPA   #'T        ; TRACE
00659A F448 26     05   F44F        BNE    SWIS14     ; NOT
00660A F44A B6     FF68    A        LDA    PCOUNT     ;
00661A F44D 27     0A   F459        BEQ    SWIS16     ; CONTINUE
00662A F44F BE     FF6C    A SWIS14 LDX    OPRND2
00663A F452 27     E3   F437        BEQ    SWIS10     ; TRACE END
00664A F454 30     1F      A        LEAX   -1,X
00665A F456 BF     FF6C    A        STX    OPRND2
00666A F459 10CE FF8F      A SWIS16 LDS    #STACK     ; TEMP STACK
00667A F45D FE     FF22    A        LDU    .SSAVE     ;
00668A F460 8D     74   F4D6        BSR    RLOAD      ; REG RESTORE
00669A F462 1F     34      A        TFR    U,S        ;
00670A F464 86     E2      A SWIS18 LDA    #$E2       ; MODE SET
00671A F466 B7     FCC0    A        STA    PTM        ;
00672A F469 8E     000D    A        LDX    #$D
00673A F46C BF     FCC2    A        STX    PTM+2      ; TIMMER SET
00674A F46F 3B                      RTI               ; CONTINUE
00675                        *
00676                        *
00677                        ********************************
00678                        *
00679                        * SWAP
00680                        *
00681A F470 8E     FF35    A SWAP   LDX    #BRKCNT    ; BREAK POINT TABLE
00682A F473 E6     80      A        LDB    ,X+        ; NUMBER
00683A F475 27     0C   F483        BEQ    SWAP4      ; NOT EXIST
00684A F477 A6     94      A SWAP2  LDA    [,X]       ; USER INSTRUCTION
00685A F479 A7     88 10   A        STA    16,X       ; SAVE
00686A F47C 86     3F      A        LDA    #SWI       ; SWI INSTRUCTION
00687A F47E A7     91      A        STA    [,X++]     ; NEXT
00688A F480                         DBNE   SWAP2
00689A F483 39               SWAP4  RTS
00690                        *
00691                        ************************************
00692                        *
00693                        * RCVER
00694                        *
00695A F484 8E     FF35    A RCVER  LDX    #BRKCNT
00696A F487 E6     80      A        LDB    ,X+        ; COUNTER
00697A F489 27     F8   F483        BEQ    SWAP4      ; NOT EXIST
00698A F48B A6     88 10   A RCVER2 LDA    16,X       ; USER INSTRUCTION
00699A F48E A7     91      A        STA    [,X++]     ; RESTORE
00700A F490                         DBNE   RCVER2
00701A F493 39                      RTS
00702                        **********
00703                        ************************
00704                        *
00705                        * MATCH
```

```
00706
00707A F494 10BE FF16     A MATCH   LDY     .PSAVE     ; USER P.C
00708A F498 8E   FF35     A MATC2   LDX     #BRKCNT    ; COUNT
00709A F49B E6   80       A         LDB     ,X+        ; TABLE
00710A F49D 10AC 81       A MATC4   CMPY    ,X++
00711A F4A0 27   06   F4A8          BEQ     MATC8      ; YEA!
00712A F4A2                         DBNE    MATC4      ; NEXT
00713A F4A5 86   FF       A MATC6   LDA     #-1
00714A F4A7 39                      RTS
00715                       *
00716A F4A8 4F              MATC8   CLRA
00717A F4A9 39                      RTS
00718                       *
00719                       *
00720                       *
00721                       ***********************************
00722                       *
00723                       * SAVE: REG SAVE
00724                       *
00725A F4AA A6   C0       A SAVE    LDA     ,U+
00726A F4AC B7   FF21     A         STA     .CSAVE
00727A F4AF A6   C0       A         LDA     ,U+
00728A F4B1 B7   FF1F     A         STA     .ASAVE
00729A F4B4 A6   C0       A         LDA     ,U+
00730A F4B6 B7   FF20     A         STA     .BSAVE
00731A F4B9 A6   C0       A         LDA     ,U+
00732A F4BB B7   FF1E     A         STA     .DSAVE
00733A F4BE AE   C1       A         LDX     ,U++
00734A F4C0 BF   FF1C     A         STX     .XSAVE
00735A F4C3 AE   C1       A         LDX     ,U++
00736A F4C5 BF   FF1A     A         STX     .YSAVE
00737A F4C8 AE   C1       A         LDX     ,U++
00738A F4CA BF   FF18     A         STX     .USAVE
00739A F4CD AE   C1       A         LDX     ,U++
00740A F4CF BF   FF16     A         STX     .PSAVE
00741A F4D2 FF   FF22     A         STU     .SSAVE
00742A F4D5 39                      RTS
00743                       *
00744                       ***********************************
00745                       *
00746                       * RLOAD: LOAD REG
00747                       *
00748A F4D6 BE   FF16     A RLOAD   LDX     .PSAVE
00749A F4D9 AF   C3       A         STX     ,--U
00750A F4DB BE   FF18     A         LDX     .USAVE
00751A F4DE AF   C3       A         STX     ,--U
00752A F4E0 BE   FF1A     A         LDX     .YSAVE
00753A F4E3 AF   C3       A         STX     ,--U
00754A F4E5 BE   FF1C     A         LDX     .XSAVE
00755A F4E8 AF   C3       A         STX     ,--U
00756A F4EA B6   FF1E     A         LDA     .DSAVE
00757A F4ED A7   C2       A         STA     ,-U
00758A F4EF B6   FF20     A         LDA     .BSAVE
00759A F4F2 A7   C2       A         STA     ,-U
00760A F4F4 B6   FF1F     A         LDA     .ASAVE
00761A F4F7 A7   C2       A         STA     ,-U
00762A F4F9 B6   FF21     A         LDA     .CSAVE
00763A F4FC A7   C2       A         STA     ,-U
00764A F4FE 39                      RTS
00765                       *
00766                       ***********************************
00767                       *
00768                       * NMI SERVICE ROUTINE
00769                       *
00770A F4FF 1F   43       A NMISRV  TFR     S,U
00771A F501 F6   FCC1     A         LDB     PTM+1      ; STATUS READ
00772A F504 86   A3       A         LDA     #%10100011 ; PRESET
00773A F506 B7   FC00     A         STA     PTM        ;
00774A F509 86   51       A         LDA     #$51
00775A F50B B7   FCF4     A         STA     ACIASC     ; SET
00776A F50E 5D                      TSTB
```

```
00777A F50F 2B    05   F516          BMI    NMIS02    ; NORMAL
00778A F511 86    80      A          LDA    #$80
00779A F513 B7    FF08    A          STA    ABFLAG    ; ABEND FLAG
00780A F516 17    FF57 F470 NMIS02   LBSR   SWAP
00781A F519 7D    FF08    A          TST    ABFLAG    ; ABEND?
00782A F51C 2B    05   F523          BMI    NMIS03    ; YES
00783A F51E 7D    FF07    A          TST    TRFLAG    ; TRACE?
00784A F521 2A    10   F533          BPL    NMIS04    ; NOT-RTI
00785A F523 10CE  FF8F    A NMIS03   LDS    #STACK    ; TEMP
00786A F527 8D    81   F4AA          BSR    SAVE      ; REG SAVE
00787A F529 7D    FF08    A          TST    ABFLAG    ; ABEND?
00788A F52C 102B  FED9 F409          LBMI   SWIS04    ; YES
00789A F530 16    FEEA F41D          LBRA   SWIS06    ; NOT-EXIT
00790A F533 3B              NMIS04   RTI              ; CONTINUE
00791                       *
00792                       *
00793                       *
00794                       *
00795                       ***********************************
00796                       *
00797                       *
00798A F534 8E    F5BB    A RDATA    LDX    #REGTBL
00799A F537 17    0136 F670          LBSR   STRNG2    ; P-
00800A F53A BF    FF04    A          STX    SCRACH
00801A F53D 8E    FF16    A          LDX    #.PSAVE
00802A F540 17    0317 F67A          LBSR   OUT4HS    ; DATA
00803A F543 BE    FF04    A          LDX    SCRACH
00804A F546 17    0127 F670          LBSR   STRNG2    ; X-
00805A F549 BF    FF04    A          STX    SCRACH
00806A F54C 8E    FF1C    A          LDX    #.XSAVE
00807A F54F 17    0128 F67A          LBSR   OUT4HS    ; DATA
00808A F552 BE    FF04    A          LDX    SCRACH
00809A F555 17    0118 F670          LBSR   STRNG2    ; Y-
00810A F558 BF    FF04    A          STX    SCRACH
00811A F55B 8E    FF1A    A          LDX    #.YSAVE
00812A F55E 17    0119 F67A          LBSR   OUT4HS    ; DATA
00813A F561 BE    FF04    A          LDX    SCRACH
00814A F564 17    0109 F670          LBSR   STRNG2    ; A-
00815A F567 BF    FF04    A          STX    SCRACH    ;
00816A F56A 8E    FF1F    A          LDX    #.ASAVE
00817A F56D 17    010C F67C          LBSR   OUT2HS    ; DATA
00818A F570 BE    FF04    A          LDX    SCRACH
00819A F573 17    00FA F670          LBSR   STRNG2    ; B-
00820A F576 BF    FF04    A          STX    SCRACH
00821A F579 8E    FF20    A          LDX    #.BSAVE
00822A F57C 17    00FD F67C          LBSR   OUT2HS
00823A F57F BE    FF04    A          LDX    SCRACH
00824A F582 17    00EB F670          LBSR   STRNG2    ; C-
00825A F585 BF    FF04    A          STX    SCRACH    ;
00826A F588 8E    FF21    A          LDX    #.CSAVE
00827A F58B 17    00EE F67C          LBSR   OUT2HS
00828A F58E BE    FF04    A          LDX    SCRACH
00829A F591 17    00DC F670          LBSR   STRNG2    ; DP-
00830A F594 BF    FF04    A          STX    SCRACH
00831A F597 8E    FF1E    A          LDX    #.DSAVE
00832A F59A 17    00DF F67C          LBSR   OUT2HS
00833A F59D BE    FF04    A          LDX    SCRACH
00834A F5A0 17    00CD F670          LBSR   STRNG2
00835A F5A3 BF    FF04    A          STX    SCRACH
00836A F5A6 8E    FF18    A          LDX    #.USAVE
00837A F5A9 17    00CE F67A          LBSR   OUT4HS    ; DATA
00838A F5AC BE    FF04    A          LDX    SCRACH
00839A F5AF 17    00BE F670          LBSR   STRNG2    ; S-
00840A F5B2 BF    FF04    A          STX    SCRACH    ;
00841A F5B5 8E    FF22    A          LDX    #.SSAVE
00842A F5B8 16    00BF F67A          LBRA   OUT4HS
00843                       *
00844A F5BB       50      A REGTBL   FCB    'P,'-,EOT
00845A F5BE       58      A          FCB    'X,'-,EOT
00846A F5C1       59      A          FCB    'Y,'-,EOT
00847A F5C4       41      A          FCB    'A,'-,EOT
```

```
00848A F5C7      42        A         FCB     'B,'-,EOT
00849A F5CA      43        A         FCB     'C,'-,EOT
00850A F5CD      44        A         FCB     'D,'P,'-,EOT
00851A F5D1      55        A         FCB     'U,'-,EOT
00852A F5D4      53        A         FCB     'S,'-,EOT
00853                      *
00854                      *
00855                      ********************************
00856                      *
00857                      * FUNCTION; PDATA
00858                      * DESCRIPTION; DUMP DATA PRINT
00859                      * INPUTS; NONE
00860                      * OUTPUTS; NONE
00861                      * SUB- OUT2HS,OUT4HS
00862                      *
00863A F5D7 CE   FF24      A DDATA  LDU     #DMPCNT   ;
00864A F5DA E6   C0        A        LDB     ,U+       ; TABLE
00865A F5DC 1F   31        A PDATA2 TFR     U,X
00866A F5DE 17   0099 F67A          LBSR    OUT4HS    ; ADDRESS
00867A F5E1 AE   C1        A        LDX     ,U++
00868A F5E3 17   0096 F67C          LBSR    OUT2HS    ; DATA
00869A F5E6                         DBNE    PDATA2
00870A F5E9 16   00D8 F6C4          LBRA    CRLF      ; CR,LF
00871                      *
00872                      *
00873                      ********************************
00874                      *
00875                      * SET OPERAND1,"2,"3
00876                      *
00877A F5EC CE   FF6A      A SETOPR LDU     #OPRND1   ; BUFFER
00878A F5EF 7F   FF68      A        CLR     PCOUNT    ; CLEAR
00879A F5F2 7F   FF69      A SETOP2 CLR     COUNT     ; CHARACTER COUNT
00880A F5F5 8E   FF04      A        LDX     #SCRACH   ; TEMP INITIAL
00881A F5F8 6F   84        A        CLR     ,X
00882A F5FA 6F   01        A        CLR     1,X
00883A F5FC 17   00C1 F6C0 SETOP4   LBSR    ECHO      ; GET OPRAND
00884A F5FF B7   FF67      A        STA     LSTCHR    ; LAST CHARACTER
00885A F602 81   2E        A        CMPA    #'.       ; EXIT?
00886A F604 1027 FA67 F06F          LBEQ    COMAND    ;
00887A F608 17   0138 F743          LBSR    HEXCHK    ; HEX CHECK
00888A F60B 26   17   F624          BNE     SETOP8    ; NOT-TERMINATOR?
00889A F60D 17   00A7 F6B7          LBSR    ASBIN     ; YES
00890A F610 C6   04        A        LDB     #4
00891A F612 68   01        A SETOP6 ASL     1,X
00892A F614 69   84        A        ROL     ,X
00893A F616 25   40   F658          BCS     ERR1      ;
00894A F618                         DBNE    SETOP6    ;
00895A F61B AB   01        A        ADDA    1,X       ; MERGE
00896A F61D A7   01        A        STA     1,X
00897A F61F 7C   FF69      A        INC     COUNT
00898A F622 20   D8   F5FC          BRA     SETOP4    ;
00899A F624 81   2C        A SETOP8 CMPA    #',       ; DELIMITOR?
00900A F626 26   17   F63F          BNE     SETOP9    ; NOT
00901A F628 7D   FF69      A        TST     COUNT
00902A F62B 27   2B   F658          BEQ     ERR1
00903A F62D 10AE 84        A        LDY     ,X
00904A F630 10AF C1        A        STY     ,U++      ; STORE
00905A F633 B6   FF68      A        LDA     PCOUNT
00906A F636 81   07        A        CMPA    #7
00907A F638 24   1E   F658          BCC     ERR1      ; OPERAND OVER
00908A F63A 7C   FF68      A        INC     PCOUNT
00909A F63D 20   B3   F5F2          BRA     SETOP2    ; CONTINUE
00910A F63F 81   0D        A SETOP9 CMPA    #CR
00911A F641 26   10   F653          BNE     STOP12    ;
00912A F643 7D   FF69      A        TST     COUNT     ;
00913A F646 27   09   F651          BEQ     STOP10    ; EMPTY
00914A F648 10AE 84        A        LDY     ,X
00915A F64B 10AF C4        A        STY     ,U
00916A F64E 7C   FF68      A        INC     PCOUNT    ;
00917A F651 20   75   F6C8 STOP10   BRA     CRLF2     ;
00918A F653 81   20        A STOP12 CMPA    #SP       ;
```

```
00919A F655 26    01    F658        BNE    ERR1      ; NOT TERMONATOR
00920A F657 39                       RTS
00921                           *
00922                           *
00923                           *
00924                           *********************************
00925                           *
00926A F658 8E    F665     A ERR1   LDX    #ER1MSG
00927A F65B 20    03    F660        BRA    ERR3
00928A F65D 8E    F667     A ERR2   LDX    #ER2MSG   ;
00929A F660 8D    0E    F670 ERR3   BSR    STRNG2    ;
00930A F662 7E    F06F     A        JMP    COMAND
00931                           *
00932A F665       3F       A ER1MSG FCB    '?,EOT
00933A F667       57       A ER2MSG FCC    'WHAT?'
00934A F66C       04       A        FCB    EOT
00935                           *
00936                           *
00937                           ******************************
00938                           *
00939                           * I/O ROUTINE
00940                           *
00941                           ******************************
00942                           *
00943                           * STRING OUT
00944                           *
00945A F66D BD    F6C4     A STRNG  JSR     CRLF
00946A F670 A6    80       A STRNG2 LDA    ,X+       ; GET
00947A F672 81    04       A        CMPA   #EOT      ; END?
00948A F674 27    49    F6BF        BEQ    ASBIN2    ; YES
00949A F676 8D    55    F6CD        BSR    WRCHR     ;
00950A F678 20    F6    F670        BRA    STRNG2
00951                           *
00952                           *
00953                           ******************************
00954                           *
00955                           * PRINT 4 HEX CHARACTERS, SPACE (X)
00956                           *
00957A F67A 8D    04    F680 OUT4HS BSR    OUT2H     ; PRINT FIRST 2 HEX CHARS
00958A F67C 8D    02    F680 OUT2HS BSR    OUT2H     ; PRINT HEX CHARS
00959A F67E 20    4B    F6CB        BRA    SPACE     ; PRINT SPACE
00960                           *
00961                           ******************************
00962                           *
00963                           * OUTPUT 2 HEX CHARS (X)
00964                           *
00965A F680 A6    80       A OUT2H  LDA    ,X+       ; GET CHAR TO PRINT
00966A F682 34    02       A CHEXL  PSHS   A         ; SAVE
00967A F684 44                      LSRA
00968A F685 44                      LSRA
00969A F686 44                      LSRA
00970A F687 44                      LSRA
00971A F688 8D    02    F68C        BSR    BINAS     ;
00972A F68A 35    02       A        PULS   A         ;
00973             F68C     A CHEXR  EQU    *
00974                           *
00975                           *********************************
00976                           *
00977                           * BINARY TO ASCII & OUTPUT
00978                           *
00979A F68C 84    0F       A BINAS  ANDA   #$0F      ; MSK
00980A F68E 8B    90       A        ADDA   #$90
00981A F690 19                      DAA
00982A F691 89    40       A        ADCA   #$40      ;
00983A F693 19                      DAA
00984A F694 20    37    F6CD        BRA    WRCHR     ; OUT
00985                           *
00986                           *********************************
00987                           *
00988A F696 8D    08    F6A0 WORD   BSR    BYTE      ; 1ST BYTE
00989A F698 34    02       A        PSHS   A         ; SAVE
```

```
00990A F69A 8D   04   F6A0        BSR    BYTE      ; 2ND
00991A F69C 1F   89      A        TFR    A,B
00992A F69E 35   82      A        PULS   A,PC
00993                        *
00994                        *
00995                        *
00996A F6A0 8D   13   F6B5 BYTE   BSR    INPUT
00997A F6A2 C6   10      A        LDB    #16
00998A F6A4 3D                    MUL
00999A F6A5 34   04      A        PSHS   B         ; SAVE
01000A F6A7 8D   0C   F6B5        BSR    INPUT
01001A F6A9 AB   E0      A        ADDA   ,S+
01002A F6AB 34   02      A        PSHS   A
01003A F6AD BB   FF06    A        ADDA   CHKSUM
01004A F6B0 B7   FF06    A        STA    CHKSUM
01005A F6B3 35   82      A        PULS   A,PC
01006                        *
01007                        *
01008                        ********************************
01009                        *
01010                        *
01011A F6B5 8D   28   F6DF INPUT  BSR    RDCHR     ; ONE CHARACTER GET
01012                        *
01013                        ********************************
01014                        *
01015                        * ASCII TO BINARY
01016                        *
01017A F6B7 80   30      A ASBIN  SUBA   #'0
01018A F6B9 81   0A      A        CMPA   #10
01019A F6BB 2B   02   F6BF        BMI    ASBIN2
01020A F6BD 80   07      A        SUBA   #7
01021A F6BF 39             ASBIN2 RTS
01022                        *
01023                        *
01024                        ********************************
01025                        *
01026                        * CONSOL IN & ECHO
01027                        *
01028A F6C0 8D   1D   F6DF ECHO   BSR    RDCHR     ; IN
01029A F6C2 20   09   F6CD        BRA    WRCHR     ;
01030                        *
01031                        ********************************
01032                        *
01033A F6C4 86   0D      A CRLF   LDA    #CR
01034A F6C6 8D   05   F6CD        BSR    WRCHR     ; OUT
01035A F6C8 86   0A      A CRLF2  LDA    #LF
01036A F6CA                       SKIP2  ;         SKIP
01037                        *
01038                        ********************************
01039                        *
01040A F6CB 86   20      A SPACE  LDA    #SP
01041                        *
01042                        ********************************
01043                        *
01044A F6CD 8D   5D   F72C WRCHR  BSR    CO        ; OUT
01045A F6CF 81   0D      A        CMPA   #CR
01046A F6D1 26   0B   F6DE        BNE    WRCH4
01047A F6D3 34   10      A        PSHS   X         ; SAVE
01048A F6D5 8E   0100    A        LDX    #256
01049A F6D8 30   1F      A WRCHR2 LEAX   -1,X
01050A F6DA 26   FC   F6D8        BNE    WRCHR2
01051A F6DC 35   10      A        PULS   X         ;
01052A F6DE 39             WRCH4  RTS
01053                        *
01054                        ********************************
01055                        *
01056A F6DF 8D   41   F722 RDCHR  BSR    CI
01057A F6E1 84   7F      A        ANDA   #$7F      ; PARITY
01058A F6E3 39                    RTS
01059                        *
01060                        ********************************
```

```
01061
01062
01063A F6E4 B6    FCF4     A BRKCHK LDA    ACIASC
01064A F6FE 85    10       A        BITA   #$10      ; CHECK IF FRAMING ERROR
01065A F6E9 26    19    F704        BNE    BRKC07
01066A F6EB 47                      ASRA
01067A F6EC 24    11    F6FF        BCC    BRKC04    ; NOT
01068A F6EE B6    FCF5     A        LDA    ACIADT    ;
01069A F6F1 84    7F       A        ANDA   #$7F      ;
01070A F6F3 81    17       A        CMPA   #CNTW     ; WAIT?
01071A F6F5 26    08    F6FF        BNE    BRKC04    ; NOT
01072A F6F7 8D    29    F722 BRKC02 BSR    CI        ; YES
01073A F6F9 81    0D       A        CMPA   #CR
01074A F6FB 27    02    F6FF        BEQ    BRKC04    ; YES
01075A F6FD 20    F8    F6F7        BRA    BRKC02    ; CONTINUE
01076A F6FF 4F               BRKC04 CLRA
01077A F700 39                      RTS
01078                        *
01079A F701 4F               BRKC06 CLRA
01080A F702 4C                      INCA
01081A F703 39                      RTS
01082                        *
01083A F704 86    03       A BRKC07 LDA    #3        ; ACIA MASTER RESET
01084A F706 B7    FCF4     A        STA    ACIASC
01085A F709 86    51       A        LDA    #$51
01086A F70B B7    FCF4     A        STA    ACIASC
01087A F70E 34    10       A        PSHS   X
01088A F710 8E    4000     A        LDX    #$4000
01089A F713 30    1F       A DLY    LEAX   -1,X
01090A F715 26    FC    F713        BNE    DLY
01091A F717 35    10       A        PULS   X
01092A F719 B6    FCF4     A        LDA    ACIASC    ; ACIA STATUS SIDE
01093A F71C 85    10       A        BITA   #$10
01094A F71E 26    E4    F704        BNE    BRKC07
01095A F720 20    DF    F701        BRA    BRKC06
01096                        *
01097                        *
01098                        **********************************
01099                        *
01100A F722 B6    FCF4     A CI     LDA    ACIASC    ; RECEIVE?
01101A F725 47                      ASRA
01102A F726 24    FA    F722        BCC    CI        ; NO
01103A F728 B6    FCF5     A        LDA    ACIADT    ; YES
01104A F72B 39                      RTS
01105                        *
01106                        **********************************
01107                        *
01108A F72C 34    04       A CO     PSHS   B         ; SAVE
01109A F72E F6    FCF4     A CO2    LDB    ACIASC    ; BUSY?
01110A F731 C5    02       A        BITB   #2
01111A F733 27    F9    F72E        BEQ    CO2       ; YES
01112A F735 B7    FCF5     A        STA    ACIADT    ; SEND DATA
01113A F738 35    84       A        PULS   B,PC
01114                        *
01115                        **********************************
01116                        *
01117A F73A 1A    01       A SETC   ORCC   #1
01118A F73C 39                      RTS
01119                        *
01120A F73D 1C    FE       A RSTC   ANDCC  #$FE
01121A F73F 39                      RTS
01122A F740 1A    08       A SETM   ORCC   #8
01123A F742 39                      RTS
01124                        *
01125                        *
01126                        **********************************
01127                        *
01128                        * HEX CHECK
01129                        * HEX=Z,ALPH=M,NOT=C
01130                        *
01131                        *
```

```
01132A F743 81    30          A HEXCHK CMPA   #'0
01133A F745 2B    F3     F73A          BMI    SETC
01134A F747 81    39          A        CMPA   #'9
01135A F749 2F    0E     F759          BLE    HEXC2
01136A F74B 81    41          A        CMPA   #'A
01137A F74D 2B    EB     F73A          BMI    SETC
01138A F74F 81    46          A        CMPA   #'F
01139A F751 2F    08     F75B          BLE    HEXC4
01140A F753 81    5A          A        CMPA   #'Z
01141A F755 2F    0A     F761          BLE    HEXC6
01142A F757 20    E1     F73A          BRA    SETC
01143A F759 5F                  HEXC2  CLRB
01144A F75A 39                         RTS
01145A F75B 5F                  HEXC4  CLRB                 ; C=0
01146A F75C C6    FF          A        LDB    #-1
01147A F75E 1A    04          A        ORCC   #4            ; Z=1
01148A F760 39                         RTS
01149A F761 5F                  HEXC6  CLRB                 ; C=0
01150A F762 C6    FF          A        LDB    #-1
01151A F764 1C    FB          A        ANDCC  #$FB
01152A F766 39                         RTS
01153                           *
01154                           *
01155                           ***********************************
01156                           *
01157                           *
01158A F767 CE    FF00        A TBLCLR LDU    #BEGADR   ; TABLE START ADR
01159A F76A C6    8F          A        LDB    #STACK-BEGADR ; LENGTH
01160A F76C 6F    C0          A TBLCL2 CLR    ,U+
01161A F76E                            DBNE   TBLCL2
01162A F771 39                         RTS
01163                           *
01164                           *
01165                           ***********************************
01166                           *
01167                           *
01168             F772        A CMDTBL EQU    *
01169A F772      41          A        FCB    'A
01170A F773      F0AC        A        FDB    ACMND
01171                           *
01172A F775      42          A        FCB    'B
01173A F776      F0AC        A        FDB    BCMND
01174                           *
01175A F778      43          A        FCB    'C
01176A F779      F0B1        A        FDB    CCMND
01177                           *
01178A F77B      44          A        FCB    'D
01179A F77C      F0DE        A        FDB    DCMND
01180                           *
01181A F77E      45          A        FCB    'E
01182A F77F      F139        A        FDB    ECMND
01183                           *
01184A F781      49          A        FCB    'I
01185A F782      F16E        A        FDB    ICMND
01186                           *
01187A F784      4C          A        FCB    'L
01188A F785      F19E        A        FDB    LCMND
01189                           *
01190A F787      4D          A        FCB    'M
01191A F788      F238        A        FDB    MCMND
01192                           *
01193A F78A      4E          A        FCB    'N
01194A F78B      F300        A        FDB    NCMND
01195                           *
01196A F78D      50          A        FCB    'P
01197A F78E      F300        A        FDB    PCMND
01198                           *
01199A F790      51          A        FCB    'Q
01200A F791      F300        A        FDB    QCMND
01201                           *
01202A F793      52          A        FCB    'R
```

```
01203A F794        F300    A           FDB     RCMND
01204                        *
01205A F796        53      A           FCB     'S
01206A F797        F30F    A           FDB     SCMND
01207                        *
01208A F799        54      A           FCB     'T
01209A F79A        F32F    A           FDB     TCMND
01210                        *
01211A F79C        55      A           FCB     'U
01212A F79D        F363    A           FDB     UCMND
01213                        *
01214A F79F        58      A           FCB     'X
01215A F7A0        F363    A           FDB     XCMND
01216                        *
01217A F7A2        59      A           FCB     'Y
01218A F7A3        F363    A           FDB     YCMND
01219                        *
01220A F7A5        5A      A           FCB     'Z
01221A F7A6        F363    A           FDB     ZCMND
01222                        *
01223              F7A8    A CMDEND    EQU     *
01224                        *
01225                        *****************************
01226                        *
01227A F7A8        3E      A PROMPT    FCB     '>,EOT
01228A F7AA        41      A ABORT     FCC     'ABORT'
01229A F7AF        04      A           FCB     EOT
01230                        *
01231A F7B0        53      A SWINS     FCC     'SWI-'
01232A F7B4        04      A           FCB     EOT
01233                        *
01234A F7B5        43      A CKSUM     FCC     'CHECKSUM ERROR'
01235A F7C3        04      A           FCB     EOT
01236                        *
01237A F7C4        56      A VERFY     FCC     'VERIFY ERROR'
01238A F7D0        04      A           FCB     EOT
01239                        *
01240A F7D1        4C      A LDEND     FCC     'LOAD END'
01241A F7D9        04      A           FCB     EOT
01242                        *
01243A F7DA        4D      A HEAD      FCC     'M6809 MONITOR'
01244A F7E7        04      A           FCB     EOT
01245                        *
01246A F7FE                            ORG     $F7FE
01247A F7FE        F000    A           FDB     POWUP
01248                        *
01249                        *
01250              F000    A           END     POWUP
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

附录15　6809某些实用程序

以下所给的6809实用程序均以宏汇编程序写出。

1. **使用正整数的8位恢复法除法**

附表15.1 正整数8位恢复法除法程序清单

```
* THIS POSITION-INDEPENDENT CODE SUBROUTINE GENERATES THE
* QUOTIENT OF TWO 8-BIT POSITIVE OPERANDS.
* THE SUBROUTINE USES A, B, AND CCR.  ROUTINE REQUIRES 3N/2
* ADDITIONS, WHERE N=8.  (RESTORING METHOD IS USED)
* STACK PICTURE ON ENTRY
*        U+0        OLD STACK MARK
*        U-1        DIVISOR
*        U-2        DIVIDEND
*        U-3        COUNT (=-8 INITIALLY)

* STACK PICTURE ON EXIT
*        U+0        OLD STACK MARK
*        U-1        QUOTIENT
*        U-2        REMAINDER
*        U-3        UNUSED
* CALLING ROUTINE SAVES STACK SPACE AND STACK MARK POINTER
* AND PASSES PARAMETERS
* CALLING ROUTINE SHOULD DO THE FOLLOWING
*        PSHS       U
*        TFR        S,U
*        LEAS       -3,S
*         .
*         .                              STACK PARAMETERS
*         .
*        LBSR       RSDVSR
*        TFR        U,S
*        PULS       U,PC
* SUBROUTINE BODY FOR 8-BIT RESTORING DIVISION
RSDVSR   PSHS       B,A,CCR              PRESERVE MPU STATUS
         CLRA
         LDB        DIVD,U               GET DIVIDEND
* ADJUST DIVIDEND AND QUOTIENT
* TRIAL SUBTRACTION: DIVISOR .LT.  DIVIDEND ?
LOOP     ASLB                            ALIGN DIVIDEND AND QUOTIENT
         ROLA
         SUBA       DVSR,U
         BMI        NEGA                 DIVISOR .LT. DIVIDEND?
         INCB                            YES, INC QUOTIENT
         BRA        ENDCHK               SKIPOVER
* NO, RESTORE DIVIDEND : TRIAL FAILED
NEGA     ADDA       DVSR,U               ADD DIVISOR BACK
* YES, NOW CHECK FOR EIGHT TRIALS
ENDCHK   INC        CNT,U
         BLT        LOOP                 8 TRIALS?
         STA        RMDR,U               STACK RESULTS
         STB        QUOT,U
DONE     PULS       CCR,A,B,PC           RETURN
* END OF SUBROUTINE
* STACK OFFSET VALUES FOR PARAMETERS
CNT      EQU        -3                   COUNT
DVSR     EQU        -1                   DIVISOR
DIVD     EQU        -2                   DIVIDEND
RMDR     EQU        -2                   REMAINDEF
QUOT     EQU        -1                   QUOTIENT
```

2. 文本字符串检索程序

附表15.2 文本字符串检索程序清单

```
00032                          *
00033                          * SEARCH LOOKS FOR A PARTICULAR TEXT STRING
00034                          *      IN A BLOCK OF DATA.
00035                          *      RETURNS Z=1 IFF FOUND.
00036                          *      X POINTS AT NEXT CHAR PAST STRING.
00037                          *
00038 1443 30   8D  0038  9 START  LEAX BLOCK,PCR  DATA BLOCK START ADDR
00039 1447 33   8D  0061  9        LEAU END,PCR  DATA BLOCK END ADDR
00040 144B 31   8D  005E  9        LEAY STRING,PCR  ADDR OF STRING TO BE FOUND
00041 144F C6   05        2        LDB  #LENGTH
00042 1451 8D   02        7        BSR  SEARCH
00043 1453 20   FE        3        BRA  *

00045 1455 34   74       11 SEARCH PSHS U,Y X,B
00046                          *
00047                          * (SP+0) = LENGTH
00048                          * (SP+1) = RESTART BLOCK SEARCH (H)
00049                          *          RESTART BLOCK SEARCH (L)
00050                          * (SP+3) = STRING (H)
00051                          *          STRING (L)
00052                          * (SP+5) = END (H)
00053                          *          END (L)
00054 1457 AE   61        6 AGAIN  LDX  1,S
00055 1459 10AE 63        7        LDY  3,S        RESET STRING PTR
00056 145C E6   E4        4        LDB  0,S        RESET STRING LENGTH
00057                          * THIS LOOP SEARCHES AFTER MISMATCH
00058 145E AC   65        7 LOOP1  CMPX 5,S        END OF DATA?
00059 1460 2E   1A        3        BGT  EXIT       IF YES, EXIT NOT FOUND
00060 1462 A6   80        6        LDA  ,X+        GET BYTE AND INC
00061 1464 AF   61        6        STX  1,S        STORE RESTART LOCATION
00062 1466 A1   A4        4        CMPA 0,Y        SAME AS STRING?
00063 1468 26   F4        3        BNE  LOOP1      BRANCH IF NOT
00064 146A 31   21        5        LEAY 1,Y        POINT TO 2ND CHAR
00065 146C 5A             2        DECB
00066 146D 27   0D        3        BEQ  _EXIT      FOR 1-BYTE SEARCH
00067                          * THIS LOOP SEARCHES AFTER MATCH
00068 146F AC   65        7 LOOP2  CMPX 5,S        END OF DATA?
00069 1471 2E   09        3        BGT  EXIT       IF YES, EXIT NOT FOUND
00070 1473 A6   80        6        LDA  ,X+        GET BYTE AND INC
00071 1475 A1   A0        6        CMPA ,Y+        SAME AS STRING?
00072 1477 26   DE        3        BNE  AGAIN      IF NO, START OVER
00073 1479 5A             2        DECB            DONE?
00074 147A 26   F3        3        BNE  LOOP2      IF NO, KEEP GOING
00075 147C 32   67        5 EXIT   LEAS 7,S        CLEAN UP STACK
00076 147E 39             5        RTS

00078 147F      54          BLOCK  FCC  /THIS IS A BLOCK OF DATIVE/
00079 1499      44                 FCC  /DATA TO BE SEARCHED./
00080           14AC        END    EQU  *-1
00081 14AD      44          STRING FCC  /DATA/
00082           0005        LENGTH EQU  *-STRING
```

3. 文本缓冲区检索

附表15.3 文本缓冲区检索程序清单

```
00001                              NAM   AUTOEX
00003                              OPT   LLEN=80
00004
00005                        ******************************************
00006                        *  COMPARE STRINGS SUB
00007                        *
00008                        *  FIND AN INPUT ASCII STRING POINTED TO BY THE
00009                        *  X-REGISTER IN A TEXT BUFFER POINTED TO BY THE
00010                        *  Y-REGISTER. THE BUFFER IS TERMINATED BY A
00011                        *  BYTE CONTAINING A NEGATIVE VALUE. ON ENTRY
00012                        *  A CONTAINS THE LENGTH OF THE INPUT STRING. ON
00013                        *  EXIT, Y CONTAINS THE POINTER TO THE START
00014                        *  OF THE MATCHED STRING + 1 IFF Z IS SET. IFF Z
00015                        *  IS NOT SET THE INPUT STRING WAS NOT FOUND.
00016                        *
00017                        *  ENTRY:
00018                        *     X POINTS TO INPUT STRING
00019                        *     Y POINTS TO TEXT BUFFER
00020                        *     A LENGTH OF INPUT STRING
00021                        *  EXIT:
00022                        *     IFF Z=1 THEN Y POINTS TO MATCHED STRING + 1
00023                        *     IFF Z=0 THEN NO MATCH
00024                        *     X IS DESTROYED
00025                        *     B IS DESTROYED
00026                        *
00027                        ******************************************
00028                        *
00029 0100                         ORG   $100
00030 0100  E6 A0      6   CMPSTR  LDB   ,Y+        GET BUFFER CHARACTER
00031 0102  2A 01      3           BPL   CMP1       BRANCH IF NOT AT BUFFER END
00032 0104  39         5           RTS              NO MATCH, Z=0
00033 0105  E1 84      4   CMP1    CMPB  ,X         COMPARE TO FIRST STRING CHAR
00034 0107  26 F7      3           BNE   CMPSTR     BRANCH ON NO COMPARE
00035                      * SAVE STATE SO SEARCH CAN BE RESUMED IF IT FAILS
00036 0109  34 32      9           PSHS  A,X,Y
00037 010B  30 01      5           LEAX  1,X        POINT X TO NEXT CHAR
00038 010D  4A         2   CMP2    DECA             ALL CHARS COMPARE?
00039 010E  27 0C      3           BEQ   CMPOUT     IF SO, IT'S A MATCH, Z=1
00040 0110  E6 A0      6           LDB   ,Y+        GET NEXT BUFFER CHAR.
00041 0112  2B 08      3           BMI   CMPOUT     BRANCH IF BUFFER END, Z=0
00042 0114  E1 80      6           CMPB  ,X+        DOES IT MATCH STRING CHAR?
00043 0116  27 F5      3           BEQ   CMP2       BRANCH IF SO
00044 0118  35 32      9           PULS  A,X,Y      SEARCH FAILED, RESTART SEARCH
00045 011A  20 E4      3           BRA   CMPSTR
00046 011C  35 B2     11   CMPOUT  PULS  A,X,Y,PC   FIX STACK, RETURN WITH Z
00047
00048            0000              END
```

4. ASCII数转换为十进制数

附表15.4 ASCII代码数转换为十进制数程序清单

```
  * CONVERTS ASCII NUMERAL TO DECIMAL.  REENTRANT CODE
  * STACK PICTURE ON ENTRY AND EXIT
  *        U+0        OLD STACK MARK
  *        U-2        ADDRESS OF ASCII CHARACTER
  *        U-4        ADDRESS OF DECIMAL NUMBER
  * SUBROUTINE USES A AND CCR
  * SUBROUTINE EXITS WITH ALL ONES IN (U-2)
  * IF INPUT IS ALPHA CHARACTER, ELSE A DECIMAL DIGIT
  *
* CALLING ROUTINE SAVES STACK SPACE, STACK MARK
* AND PASSES PARAMETERS:
*        PSHS     U                    SAVE STACK MARK POINTER
*        TFR      S,U
*        LEAS     -4,S
*         .
*         .                            STACK PARAMETERS
*         .
*        LBSR     ACDEC
*        TFR      U,S
*        PULS     U,PC

* SUBROUTINE BODY FOR CONVERSION
ACDEC    PSHS     A,CCR                STACK OLD A, CCR
         LDA      [CHAR,U]             GET ASCII INPUT
         SUBA     #'0                  MINUS ASCII ZERO
         CMPA     #10                  RESULT .LT. 10 ?
         BLO      END                  YES, NUMERIC!
         LDA      #$FF                 NO ALPHA
END      STA      [DEC,U]              SAVE IT
         PULS     CCR,A,PC             RETURN
* END OF SUBROUTINE

* STACK OFFSET VALUES FOR PARAMETERS
CHAR     EQU      2                    ASCII INPUT CHARACTER
DEC      EQU      -4.                  DECIMAL NUMBER RESULT
```

5. 递归阶乘

附表15.5 递归阶乘程序清单

```
* Enter:  X has address of last result
*         Y has address of current result
          A has or will have current number
* Exit:   same but updated
* This is the routine for calling the factorial
* subroutine. It initializes the storage in
* memory and prepares for the recursive subroutine.
* Enter:  A had the number to find factorial product
* Exit:   X has address of final result, either
               RESLT1 or RESLT2
* Calling Routine
   CALFAC  LDX    #RESLT1     /set up pointer
           LDY    #RESLT2     /set up pointer
           LDB    MAXBYT
           DECB               /MAXBYT-1
   INIT    CLR    B, X        /clear storage
           CLR    B, Y        /clear storage
           DECB               /decrement pointer
           BNE    INIT        /storage cleared?
           LDB    MAXBYT      /yes, restore B
           DECB               /MAXBYT-1
           INC    B, X        /put in 1 for first multiplication
           LBSR   FACT        /jump to Factorial subroutine
*                 .
*                 .
*                 .
* Subroutine
   FACT    CMPA   #1          /N=1
           BLE    RETURN      / return when A=1, Factorial=1
           PSHU   A           / put current number on U-stack
           DECA               /N-1

           LBSR   FACT        /save next PC on S stack
   PROD    PULU   A           / put current number in A
           LDB    MAXBYT      / Initialize loop counter
           DECB               /next factor in stack (MAXBYT-1)
           STB    BYTCNT      / store offset count in memory
   MULBYT  LDB    BYTCNT,X    / put a byte of last result in B
           CLR    BYTCNT,X    / for future purposes
           DEC    BYTCNT
           MUL                /8x8 multiply
           ADDD   BYTCNT,Y    / add new partial result to current result
           TST    BYTCNT      /BYTCNT-0=0?
           BGE    MULBYT      / loop back if BYTCNT≥0
           EXG    X, Y        / adjust addresses for exit
   RETURN  RTS                /go to PROD, use S stack
```

6. 8位不恢复法除法

附表15.6 8位不恢复法除法程序清单

```
' THIS POSITION-INDEPENDENT ROUTINE COMPUTES THE
* 8-BIT QUOTIENT OF TWO 8-BIT POSITIVE INTEGERS
* USING A NONRESTORING DIVISION ALGORITHM.
* ROUTINE USES A, B, AND CCR.
* N ADDITIONS AND SUBTRACTIONS ARE BEQUIBED WHERE
* N IS NUMBER OE BITS.

* STACK PICTURE ON ENTRY
*          U+0         OLD STACK MARK
*          U-1         COUNT ( = -8 INITIALLY)
*          U-2         DIVISOR
*          U-3         DIVIDEND
*
* STACK PICTURE ON EXIT
*          U+0         OLD STACK MARK
*          U-1         REMAINDER
*          U-2         QUOTIENT
*
* CALLING ROUTINE SAVES STACK SPACE, STACK MARK
* AND PASSES PARAMETERS
*          PSHS        U
*          TFR         S,U
*          LEAS        -3,S
*
*                                       STACK PARAMETERS
*
*          LBSR        NRDV
*          TFR         U,S
*          PULS        U,PC
*
* SUBROUTINE BODY FOR 8-BIT NONRESTORING DIVISION
*
NRDV       PSHS        B,A,CCR          STACK OLD MPU
           CLRA
           LDB         DIVD,U
           ASLB
           ROLA
           SUBA        DVSR,U
* SUBTRACT DIVISOR FROM DIVIDEND
LOOP       TSTA
           BMI         NEGA
```

```
* PARTIAL REMAINDER NOW POSITIVE, SUBTRACT DIVISOR
          ASLB
          ROLA
          INCB
          SUBA          DVSR,U
* SKIP OVER
          BRA           ENDCHK
* ADD DIVISOR SINCE PARTIAL REMAINDER BECAME NEGATIVE
NEGA      ASLB
          ROLA
          ADDA          DVSR,U
* N LOOP CHECK
ENDCHK    INC           CNT,U
          BLT           LOOP
          TSTA
          BMI           DONE
*
* END CORRECTION OF PARTIAL REMAINDER, ONE TIME ONLY
*
CORR      ADDA          DVSR,U
* STACK REMAINDER AND QUOTIENT
          STA           RMDR,U
DONE      STB           QUOT,U
          PULS          CCR,A,B,PC
* END OF SUBROUTINE BODY

* STACK OFFSET VALUES FOR COUNTER, DIVISOR, DIVIDEND
* QUOTIENT, AND REMAINDER
*
CNT       EQU           -1                  COUNT
DVSR      EQU           -2                  DIVISOR
DIVD      EQU           -3                  DIVIDEND
RMDR      EQU           -1                  REMAINDER
QUOT      EQU           -2                  QUOTIENT
```

7. 再入型16位除法

附表15.7 再入型16位除法程序清单

```
* THIS SUBROUTINE DIVIDES TWO 16-BIT POSITIVE INTEGERS
* GIVING A 16-BIT RESULT. SUCCESSIVE TRIAL SUBTRACTIONS
* ARE USED, WITH DIVIDEND RESTORATION AT EACH TRIAL IF
* C FLAG IS CLEARED. DIVISION NEED NOT BE NORMALIZED
* (.I.E., MSB = 1) AT ENTRY
* SUBROUTINE USES A, B, AND CCR
* STACK PICTURE AT ENTRY:
*         U+0       OLD STACK MARK
*         U-1       COUNTER (=1 INITIALLY)
*         U-2       MSB DIVISOR
*         U-3       LSB DIVISOR
*         U-4       MSB DIVIDEND
*         U-5       LSB DIVIDEND

* STACK PICTURE ON EXIT:
*         U-4       QUOTIENT (MSB)
*         U-5       QUOTIENT (LSB)

* CALLING ROUTINE GENERATES STACK STORAGE, SAVES STACK MARK
*         PSHS      U                   SAVE STACK MARK
*         TFR       S,U
```

```
*         LEAS      -5,S
*          .
*          .                              SET UP DIVISOR,
*          .                              DIVIDEND, AND COUNTER
*         LBSR      DIVG                  CALL DIVIDER
*          .
*          .
*          .
*         TFR       U,S                   CLEAN UP STACK
*         PULS      U,PC

* SUBROUTINE BODY FOR 16-BIT DIVISION
DIVG      PSHS      B,A,CCR               WE NEED THEM
          TST       MDVSR,U               DIVISOR NORMALIZED?
          BMI       NORMAL                YES, GO AHEAD
* NORMALIZE DIVISION TILL LEAD BIT IS 1, COUNTING LEFT SHIFTS
NNORM     INC       CNT,U
          ASL       LDVSR,U               SHIFT LSB DIVISOR
          ROL       MDVSR,U               NOW MSB
          BMI       NORMAL
          LDA       CNT,U
          CMPA      #17
          BNE       NORMAL                NORMAL?
* GET DIVIDEND IN D, CLEAR QUOTIENT
NORMAL    LDD       MDIVD,U
          CLR       MDIVD,U
          CLR       LDIVD,U
* BEGIN DIVISION BY SUCCESSIVE SUBTRACTION METHOD
* TESTING CARRY FLAG EACH TIME
DVD       SUBD      MDVSR,U               TRIAL SUBTRACTION
          BCC       SETC                  OK?
          ADDD      MDVSR,U               NO, RESTORE DIVIDEND
          ANDCC     #$FE                  CLEAR CARRY
          BRA       LINUP                 SKIP NEXT INSTRUCTION
* SET CARRY BIT
SETC      ORCC      #$01                  TRIAL SUBTRACTION OK
*LINE UP QUOTIENT, DIVIDEND, AND DIVISOR FOR NEXT TRIAL
LINUP     ROL       LDIVD,U               ADJUST QUOTIENT
          ROL       MDIVD,U               AND DIVIDEND
          LSR       MDVSR,U               ADJUST DIVISOR
          ROR       LDVSR,U
          DEC       CNT,U                 DONE?
          BNE       DVD                   NO
* RESTORE MPU REGISTERS AND EXIT
DONE      PULS      CCR,A,B,PC            YES, RETURN

* STACK OFFSET VALUES FOR COUNTER, DIVISOR, DIVIDEND
CNT       EQU       -1                    COUNT
MDVSR     EQU       -2                    MSB DIVISOR
LDVSR     EQU       -3                    LSB DIVISOR
MDIVD     EQU       -4                    MSB DIVIDEND
LDIVD     EQU       -5                    LSB DIVIDEND
```

8. 浮点加法和减法

附表15.8 浮点加法和减法程序清单

```
* THIS SUBROUTINE USES 16-BIT MANTISSAS (POSITIVE
* ONLY) AND 8-BIT BIASED EXPONENTS. THE EXPONENT RANGE
* IS 2^-128 (SMALLEST) TO 2^+127 (LARGEST). BIAS IS 200_8.
*
* BIT 7 OF THE EXPONENT IS USED FOR BIAS, NOT FOR
* ALGEBRAIC SIGN. INSTR IS POSITIVE FOR ADDITION, ZERO FOR
* SUBTRACTION. LEXP IS LOCAL STORAGE FOR LARGEST EXPONENT.
* EXPONENT OVERFLOW FLAG = $FF IN STACK. MAXEXP IS A
* CONSTANT = FF.
*
STACK PICTURE ON ENTRY AND EXIT
*          U+0       OLD STACK MARK
*          U-1       ERROR FLAG (=$FF IF OVERFLOW)
*          U-2       INSTR FLAG (1 IF ADD, 0 IF SUBTRACT)
*          U-3       LSB MANTISSA 1
*          U-4       MSB MANTISSA 1
*          U-5       LSB MANTISSA 2
*          U-6       MSB MANTISSA 2
*          U-7       EXPONENT 1
*          U-8       EXPONENT 2
*          U-9       LSB MANTISSA RESULT
*          U-10      MSB MANTISSA RESULT
*          U-11      EXPONENT RESULT
*          U-12      TRIAL LARGEST EXPONENT
*
* CALLING ROUTINE
*          PSHS      U
*          TFR       S,U
*          LEAS      -12,S
*           .
*           .                        STACK PARAMETERS
*           .
*          LBSR      FPASSR
*          TFR       U,S
*          PULS      U,PC
*
*
* SUBROUTINE BODY
*
FPASSR     PSHS      X,A,B,CCR
           LDA       EXP2,U
           LDB       EXP1,U
           STA       LEXP,U
           SUBB      EXP2,U
           BLS       EQEXM
           LDA       EXP1,U
           STA       LEXP,U
* ADJUST SMALLER EXPONENT BEFORE MANTISSA OPERATION
EQEXM      TSTB
EQEXP      BGE       EGEZ
* EXP1 SMALLER, INCREASE IT; DECREASE MANTISSA
           LSR       MMAN1,U
           ROR       LMAN1,U
           INCB
           BRA       EQEXP
EGEZ       BEQ       ASMAN
* EXP2 SMALLER, INCREASE IT; DECREASE MANTISSA
           LSR       MMAN2,U
           ROR       LMAN2,U
```

```
         DECB
         BRA      EQEXP
* EXPONENTS NOW EQUAL! ADD OR SUBTRACT ? CHECK INSTR.
ASMAN    LDD      MMAN1,U
         TST      INSTR,U
         BLT      SUBINS
         ADDD     MMAN2,U                ADD MANTISSA
         BRA      CONT
SUBINS   SUBD     MMAN2,U                SUBTRACT MANTISSA
*        STORE RESULT MANTISSA IN STACK
CONT     STD      LMANR,U
         LDB      LEXP,U
* MANTISSA ERROR CHECK
MOUA     BVC      ZREAD
         LSR      MMANR,U
         ROR      LMANR,U
         INCB
* EXPONENT TOO LARGE ?
         CMPB     MAXEXP
         BGT      ERROR
         BRA      RETURN
* EXPONENT TOO SMALL ?
ZREAD    LDX      MMANR,U
         CMPX     #00
         BNE      RENORM
* SET EXP OF RESULT TO ZERO, YES, LEAVE SUBROUTINE, ALL DONE
         CLRB
         BRA      RETURN
* EXPONENT OF RESULT NOT ZERO
* ADJUST MANTISSA OF RESULT BETWEEN 1/2 AND 1
RENORM   CMPX     #01
         BLT      RETURN
* INCREASE MANTISSA OF RESULT SO MSB = 1
         ASL      LMANR,U
         ROL      MMANR,U
* DECREASE EXPONENT OF RESULT
         DECB
         BLT      ERROR
         BRA      RENORM
* EXPONENT OF RESULT OVERFLOWED, SET OVERFLOW FLAG
ERROR    LDA      MAXEXP
         STA      ERRF,U
         STB      EXPR,U
RETURN   PULS     CCR,A,B,X,PC
* END OF SUBROUTINE
*
*
* STACK OFFSET VALUES FOR PARAMETERS
ERRF     EQU      -1                     ERROR FLAG
INSTR    EQU      -2                     ADD/SUBTRACT FLAG
LMAN1    EQU      -3                     LSB MANTISSA 1
MMAN1    EQU      -4                     MSB MANTISSA 1
LMAN2    EQU      -5                     LSB MANTISSA 2
MMAN2    EQU      -6                     MSB MANTISSA 2
EXP1     EQU      -7                     EXPONENT 1
EXP2     EQU      -8                     EXPONENT 2
LMANR    EQU      -9                     LSB RESULT MANTISSA
MMANR    EQU      -10                    MSB RESULT MANTISSA
EXPR     EQU      -11                    EXPONENT RESULT
LEXP     EQU      -12                    TEMPORARY LARGEST EXP.
MAXEXP   EQU      $FF                    MAX EXP VALUE
```

9. 16×16位的乘法

附表15.9 16×16位乘法程序清单

16 × 16 MULTIPLY

```
00230                                   *****************************************
00231                                   *
00232                                   *  MULTIPLY TWO 16-BIT POSITIVE VALUES
00233                                   *  TO GENERATE A 32-BIT PRODUCT.
00234                                   *  AT TERMINATION, BOTH INPUT VALUES
00235                                   *  AND THE RESULT WILL BE IN USER  *
00236                                   *  STACK.
00237                                   *
00238                                   *  (A:B) X (C:D) =          BDH:BDL
00239                                   *                  +    BCH :BCL
00240                                   *                  +    ADH:ADL
00241                                   *                  + ACH:ACL
00242                                   *                   -------------------
00243                                   *  SETUP:          3 LN,  10 BY,   10 CY
00244                                   *  OPERATION:     25 LN,  46 BY,  154 CY
00245                                   *  TOTAL:         28 LN,  56 BY,  164 CY
00246                                   *****************************************
00247

00249 1185  8E    11BF  3 ABC   LDX   #AA   POINTER TO A (MS BYTE)
00250 1188  108E  11C1  4       LDY   #BB
00251 118C  CE    11C3  3       LDU   #C    ADDRESS OF PRODUCT

00253 118F  6F    C4    6       CLR   0,U
00254 1191  6F    41    7       CLR   1,U
00255 1193  A6    01    5       LDA   1,X   : #A LS BYTE
00256 1195  E6    21    5       LDB   1,Y   : #B LS BYTE
00257 1197  3D         11       MUL
00258 1198  ED    42    6       STD   2,U
00259 119A  A6    84    4       LDA   0,X   : #A MS BYTE
00260 119C  E6    21    5       LDB   1,Y   : #B LS BYTE
00261 119E  3D         11       MUL
00262 119F  E3    41    7       ADDD  1,U
00263 11A1  ED    41    6       STD   1,U
00264 11A3  24    02    3       BCC   AB1
00265 11A5  6C    C4    6       INC   0,U
00266 11A7  A6    01    5 AB1   LDA   1,X   : #A LS BYTE
00267 11A9  E6    A4    4       LDB   0,Y   : #B MS BYTE
00268 11AB  3D         11       MUL
00269 11AC  E3    41    7       ADDD  1,U
00270 11AE  ED    41    6       STD   1,U
00271 11B0  24    02    3       BCC   AB2
00272 11B2  6C    C4    6       INC   0,U
00273 11B4  A6    84    4 AB2   LDA   0,X   : #A MS BYTE
00274 11B6  E6    A4    4       LDB   0,Y   : #B MS BYTE
00275 11B8  3D         11       MUL
00276 11B9  E3    C4    6       ADDD  0,U
00277 11BB  ED    C4    5       STD   0,U

00279 11BD  20    FE    3       BRA   *

00281 11BF        03E8    AA    FDB   1000
00282 11C1        01F4    BB    FDB   500
00283 11C3        0000    C     FDB   0,0
```

10. 负数处理法（为16×16位乘法用）

本程序可把任意负数转换为 2 的补数。

附表15.10 负数处理程序清单

Prepares values for the 16 X 16 Multiply in Example 5.14. Converts any negative value to its 2's complement. NEGFLG will be zero if both values are positive or both negative. So, upon return if NEGFLG is not zero, the product must be converted to its 32-bit 2's complement. (Note that the only complement operation available to the 6809 is the 8-bit 1's complement.)

```
        CLR     NEGFLG    / clear flag
        TST     AA        / test MSB of first value
        BGE     CHECK2    / branch if positive (or zero)
        INC     NEGFLG    / increment NEGFLG
        LDD     ZERO      / begin 2's complement
        SUBD    AA        /     subtract AA from 0
        STD     AA        /     store result in #AA
CHECK2  TST     BB        / test MSB of second value
        BGE     ENDCHK    / branch if positive
        DEC     NEGFLG    / decrement NEGFLG
        LDD     ZERO      / begin 2's complement
        SUBD    BB
        STD     BB

ENDCHK  JSR     ABC       / jump to 16X16 Multiply

        TST     NEGFLG    / if 0, Z=1
        BEQ     ENDSGN    / branch if 0, result positive
        LDX     #C        / load address of result
        COM     ,X+       / take 1's complement of 32-bit
        COM     ,X+       /      result, 1 byte at a time
        COM     ,X+       / starting with LSB
        COM     ,X+
LOOP    INC     ,-X       / add 1 to LSB for 2's complement
        BCC     ENDSGN    / branch if carry clear
        CMPX    #C        / compare X with original address
        BNE     LOOP      / branch to adjust other bytes
ENDSGN  BRA     *         / end of routine
ZERO    EQU     $0        / constant, used for 2's complement
NEGFLG  NOP
```

11. 递归法阶乘（32位精度）

附表15.11 递归法阶乘程序清单

```
      *
      * FAC IS THE RECURSIVE ROUTINE TO CALCULATE FACTORIAL
      *      TO 32-BIT ACCURACY (N! = N * N-1!).
      *
      *     ENTRY: ACCB = DESIRED FACTORIAL
      *            2,S - 5,S = RETURN PARAMETER AREA
      *
400C FAC    PSHS   B,A      SAVE REGISTERS TO BE USED
400E        CMPB   #1       1!?
4010        BHI    FAC1
      * WANT 1!,AND WE'VE GOT IT!
4012        LDD    #0
4015        STD    4,S
4017        LDD    #1
401A        STD    6,S
401C        PULS   A,B,PC

      * WANT N!, N ALREADY SAVED
401E FAC1   DECB            N-1
401F        LEAS   -4,S     RETURN AREA
4021        BSR    FAC      GET ACCB FACTORIAL

      * NOW HAVE N-1! IN 0,S - 3,S
      * WANT TO RETURN N! IN 8,S-11,S
      * FIRST, INIT THE MS BYTES
4023        LDD    #0
4026        STD    8,S
      * NOW DO 8 X 32 MULTIPLY (32-BIT RESULT)
      * LS BYTE FIRST

4028        LDA    5,S      SAVED N
402A        LDB    3,S      RETURNED LS BYTE
402C        MUL
402D        STD    10,S     NO CARRY POSSIBLE
      * NEXT MOST SIGNIFICANT BYTE
402F        LDA    5,S      SAVED N
4031        LDB    2,S      RETURNED BYTE
4033        MUL
4034        ADDD   9,S
4036        STD    9,S
4038        BCC    FAC2
403A        INC    8,S      NO OVERFLOW POSSIBLE
      * ALMOST MS-BYTE
403C FAC2   LDA    5,S
403E        LDB    1,S
4040        MUL
4041        ADDD   8,S
      * NOTE:  OVERFLOW POSSIBLE MODULO 2**32
4043        STD    8,S
      * NOW THE MS-BYTE
4045        LDA    5,S
4047        LDB    0,S
4049        MUL
404A        ADDB   8,S
      * OVERFLOW POSSIBLE AGAIN
404C        STB    8,S
404E        LEAS   4,S
4050        PULS   A,B,PC
```

12. 实时钟

附表15.12　实时钟程序清单

```
* CLOCK ROUTINE
* USES 60 HZ PULSE INPUT TO CA1 TO GENERATE
* INTERRUPT (IRQ) OF PROCESSOR
* TIME IS KEPT IN LOCATIONS ON THE STACK
* POINTED BY U-POINTER
* STACK PICTURE:
*        U+0     60'S
*        U-1     SECONDS
*        U-2     MINUTES
*        U-3     HOURS
*        U-4     WORKSPACE
*
* CALL INIT ROUTINE TO SET UP PIA FOR THE INTERRUPT AND
* START THE REAL TIME CLOCK.
* INIT CREATES SPACE FOR TIME ON S-STACK
* U POINTS TO 60'S LOCATION.  OTHERS CAN BE ACCESSED BY
* CONSTANT OFFSETS FROM U.
INIT     PULS    X                    RECONFIGURE RETURN ADDR
         LEAU    1,S                  CREATE SPACE FOR TIME
         LEAS    -5,S
         PSHS    X                    RETURN ADDR HERE
         LDX     CLOCK,PCR            SET UP IRQ VECTOR
```

续表

```
        STX     IRQ
        LDX     #0              RESET CLOCK
        STX     -3,U
        STX     -1,U
        LDX     #PIAADDR        LOAD PIA BASE ADDR
        LDA     #07             INITIALIZE PIA
        STA     X               AND THE CLOCK STARTS!
        PULS    PC
* CLOCK ROUTINE
* ENTERED VIA 60HZ INTERRUPT, USING IRQ
CLOCK   LDA     PIADRA          CLEAR INTERRUPT BY
*                               DUMMY READ
* INCREMENT 60'S
        LDA     0,U
        ADDA    #01
        DAA
        STA     0,U
* NOW UPDATE THE CLOCK
        LDA     #$60            SET UP MAX 60',
        STA     -4,U            SECONDS & MIN VALUE
        CLRB                    POINT TO 60'S
        LBSR    UPDATE          UPDATE 60'S
        DECB
        LBSR    UPDATE          UPDATE SECONDS
        DECB
        LBSR    UPDATE          UPDATE MINUTES
        LDA     #$24            SET UP MAX HOUR VALUE
        DECB
        LBSR    UPDATE          UPDATE HOURS
        RTI
* UPDATE ROUTINE UPDATES 60'S, SECONDS, MINUTES, AND HOURS
* IF APPROPRIATE.  WHEN NO FURTHER UPDATE IS NEEDED IT
* RETURNS FROM INTERRUPT DIRECTLY.
UPDATE  LEAY    B,U
        LDA     Y
        CMPA    -4,U            IS UPDATE NEEDED?
        BNE     RETURN          NO, EXIT INTERRUPT
        CLR     Y               YES, ZERO THE TIME UNIT
        LDA     -1,Y            INCREMENT NEXT UNIT
        ADDA    #01
        DAA                     DECIMAL ADJUST
        STA     -1,Y
        PULS    PC              CHECK NEXT UNIT
RETURN  LEAS    2,S
        RTI
```

13. 增序法排序

附表15.13 增序法排序程序清单

```
PAGE 001    SEQUX    .SA:0 SEQUX

00001                                  *  SEQUENCER
00002                                  *THIS PROGRAM SEQUENCES A SERIES OF NUMBERS
00003                                  *SUCH THAT THE LARGEST NUMBER GOES TO THE
00004                                  *LOWEST ADDRESS. THE U REGISTER FUNCTIONS
00005                                  *AS BOTH A STACK POINTER AND AN INDEX
00006                                  *REGISTER. UPON ENTERING THIS SUBROUTINE,
00007                                  *U CONTAINS THE STARTING ADDRESS OF THE
00008                                  *NUMBERS TO BE SEQUENCED, AND X, THE
00009                                  *ENDING ADDRESS.
00010                                       NAM   SEQUX
00011A   2000                               ORG   $2000
00012A   2000   34     50      A            PSHS  U,X
00013A   2002   EE     62      A       L1   LDU   2, S
00014A   2004   37     04      A       L3   PULU  B
00015A   2006   E1     C4      A            CMPB  U
00016A   2008   24     06      2010         BCC   L2
00017A   200A   37     02      A            PULU  A
00018A   200C   36     06      A            PSHU  A, B
00019A   200E   20     F2      2002         BRA   L1
00020A   2010   11A3   E4      A       L2   CMPU  S
00021A   2013   26     EF      2004         BNE   L3
00022A   2015   32     64      A            LEAS  4, S
00023A   2017   39                          RTS
00024                                       END
TOTAL ERRORS 00000—00000
TOTAL WARNINGS 00000—00000
```

附录16 EXORbus总线标准

本附录将说明莫托罗拉公司的86总线的标准接线内容，该总线适用于EXORciser系统和M68-MM单板系列。

引 线	信 号	信 号 定 义
A～C (1～3)	+5 V_{DC}	+5V直流电源——系统逻辑电路使用。
D (4)	$\overline{IRQ}$	中断请求——是可"线-或"的低电平有效信号，请求主系统（MPU）产生中断序列。主系统（MPU）在识别中断请求之前，要继续正常操作，直到完成当前指令。届时，如果主系统（MPU）接受中断，以后就将开始执行中断处理序列。
E (5)	$\overline{NMI}$	非屏蔽中断——是可"线-或"的下降沿有效信号来请求主系统(MPU)产生非屏蔽中断序列。主系统（MPU）要继续正常操作完成当前指令后，才识别该中断请求。无论有何中断屏蔽状态，主系统（MPU）都将开始执行非屏蔽中断序列。
F	VMA	有效存储器地址——由当前主系统总线产生的高电平有效信号，表示

引线	信号	说明
(6)		总线上有效存储器地址存在。
H (7)	——	保留——（在早先系统设计中为±12V之地）。
J (8)	BE	总线启动——（在早期系统设计中为E信号或ϕ2信号）。系由主系统产生的高电平有效时钟信号。在$\overline{\text{MNRDY}}$（R引线）为低电平时，该信号被展宽为高电平状态。
K (9)	——	保留——（在早期系统设计中为±12V之地）。
L (10)	MEMCLK	存储器时钟——在使用新的外部设备单板时，该引线无用。如果希望在系统内同原来的系统单板兼容，那么任何新型处理机单板设计应在该引线和J引线上提供BE信号。
M (11)	$-12V_{DC}$	-12V直流电源——系统逻辑电路使用。
N (12)	$\overline{\text{BUSREQ}}$	总线请求——（早期M6800系统中为$\overline{\text{TSC}}$）。是请求存取系统总线的低电平有效的“线-或”信号。该线为低电平时将使主系统(MPU)释放数据、地址、VMA、VUA、VXA和R/W总线。BE降为低电平后，还将产生BUSGNT信号（引线15）。
P (13)	BA	总线有效——是由主系统（MPU）产生的高电平有效信号，与BS一起表示主系统（MPU）的当前状态，但是并不反应系统总线的状态（系统总线状态要通过BUSGNT引线15得到反映）。注意BA、BS状态只对M6809系统而言。

BA	BS	状态
0	0	正常（运行）
0	1	中断确认
1	0	同步确认
1	1	暂停或总线回答

引线	信号	说明
R (14)	$\overline{\text{MNRDY}}$	存储器没准备就绪——（早期EXORcisers规定为MEMRDY，即存储器准备就绪）。是同慢速存储器板进行工作的系统使用的低电平有效的“线-或”信号。当该信号为低电平时，时钟信号将同BE高电平有效信号以及BQ（如存在时）低电平无效信号一起被展宽。
S (15)	LIC	最后指令周期——（只在M6809E系统中存在）。是M6809E微处理器在每条指令的最后周期所产生的高电平有效信号。该线在BUSGNT状态期间将处在高阻抗状态。新型设计中没有推荐。
T (16)	$+12V_{DC}$	+12V直流电源——系统逻辑电路用。
U (17)	STANDBY	+5V直流备用电源（早期EXORcisers中为+12V）——该线同备用电池的存储器板一起使用。如果没有备用电池，STANDBY线不应任意作为它用。
V	$\overline{\text{PWRFAIL}}$	电源故障——备用电池的存储器单板所用的低电平有效“线-或”信号。

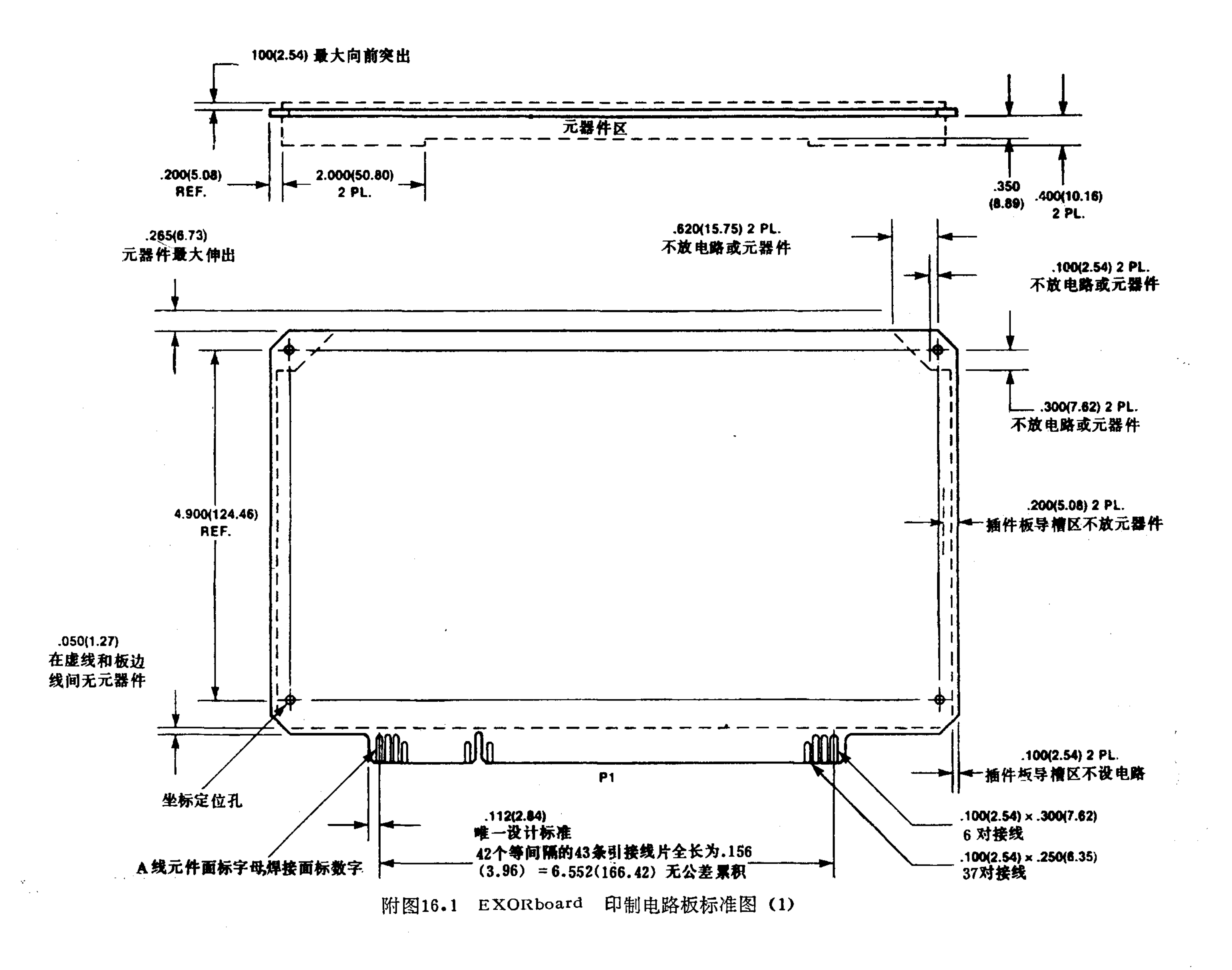
引线	信号	说明
(18)		该信号为低电平时，即保护存储器单板禁止写操作。
W (19)	$\overline{PARERR}$	奇偶错——该信号是低电平有效的“线-或”信号，正常工作时为高电平。如果存储器板在系统内使用奇偶校验电路检测出奇偶错时，该信号将变为低电平。
X～Z (20～22)	GND	地线。
$\overline{A}$ (23)	$\overline{FIRQ}$	快速中断请求——（只是M6809、M6809E系统中有）。这是低电平有效的“线-或”信号，请求主系统（MPU）产生快速中断序列。主系统将等待到当前正在执行的指令完成之后再识别该请求。届时，如果中断没被屏蔽，则主系统（MPU）就会开始中断序列。因为该中断只使返回地址和条件码进栈，所以该中断序列处理很快。
$\overline{B}$ (24)	——	保留——（早期系统设计中为参考地）
$\overline{C}$～$\overline{F}$ (25～28)	——	保留——这些引线和其对面的引线（25、26、27、28）均被保留它用。但早期的某些单板这些线作为地址选择线使用。
$\overline{H}$ (29)	$\overline{D_3}$	数据总线（第3位）——是8条双向数据线中的一条线，任务是在主系统总线和系统中所有单板之间提供双向数据传送通路。在被选中的数据传送操作期间之外，各单板上的数据总线驱动器都将处在开路或高阻抗状态。数据总线驱动器驱动数据总线时应同BE（$\phi 2$）的有效信号相重合。
$\overline{J}$ (30)	$\overline{D_7}$	数据总线（第7位）——与引线$\overline{H}$上$\overline{D_3}$说明相同。
$\overline{K}$ (31)	$\overline{D_2}$	数据总线（第2位）——与引线$\overline{H}$上$\overline{D_3}$说明相同。
$\overline{L}$ (32)	$\overline{D_6}$	数据总线（第6位）——与引线$\overline{H}$上$\overline{D_3}$说明相同。
$\overline{M}$ (33)	A14	地址总线（第14位）——是当前主系统总线驱动的16条地址线中的一条线，可对系统中任何可选地址的存储单元或外部设备接口器件进行选择。
$\overline{N}$ (34)	A13	地址总线（第13位）——与引线$\overline{M}$上A14的说明相同。
$\overline{P}$ (35)	A10	地址总线（第10位）——与引线$\overline{M}$上A14的说明相同。
$\overline{R}$ (36)	A9	地址总线（第9位）——与引线$\overline{M}$上A14的说明相同。
$\overline{S}$ (37)	A6	地址总线（第6位）——与引线$\overline{M}$上A14的说明相同。
$\overline{T}$	A5	地址总线（第5位）——与引线$\overline{M}$上A14的说明相同。

(38)

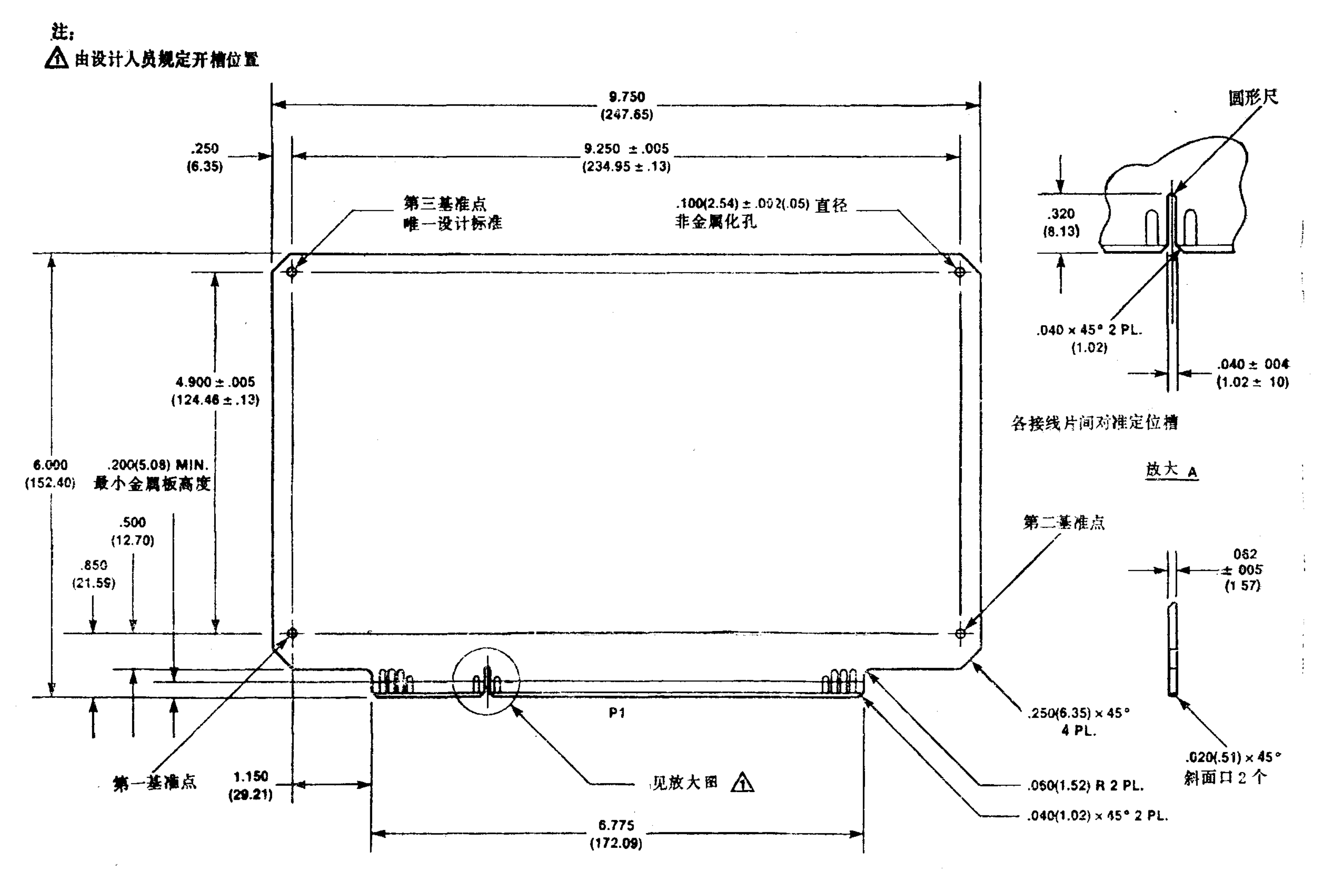
$\overline{U}$ (39)	A 2	地址总线（第 2 位）——与引线$\overline{M}$上A14的说明相同。
V (40)	A 1	地址总线（第 1 位）——与引线$\overline{M}$上A14的说明相同。
$\overline{W}$～$\overline{Y}$ (41～43)	GND	地线。

引 线	信 号	信 号 定 义
1～3	+ 5 V_{DC}	+ 5 V直流电源——系统逻辑电路使用。
4	$\overline{HALT}$	暂停——是低电平有效的可“线-或”的信号，它可在当前指令结束时暂停主系统（MPU）的工作。暂停状态可以通过BA（引线P）和BS（引线23）来表示，在该状态时不允许访问系统总线。参考$\overline{BUSREQ}$（引线N）和BUSGNT（引线15）。
5	$\overline{RESET}$	总清（复位）——是低电平有效、可“线-或”的信号，其作用是总清主系统（MPU）和其它外围器件和系统单板组件。当电源加到系统之后，该信号即启动系统的初始化程序。在系统正在操作运行时，随意按下RESET开关时都会产生$\overline{RESET}$信号，并使主系统（MPU）去执行再启动程序。
6	R/$\overline{W}$	读/写——是给系统各单板组件指派工作的主系统总线当前产生的高电平有效信号（读操作）或者低电平有效信号（写操作），表示当前主系统总线正在占用并按某方向传送数据。
7	EQ	总线定时信号——（只对M6809、M6809E系统而言）早期系统设计中定义为Q或ϕ 1 信号，该信号超前BE信号。
8,9	—	保留——（早期系统设计中为＋12V地）。
10	VUA	有效用户地址——是主系统总线当前产生的高电平有效信号，表示总线上现行的有效用户地址。在BUSGNT（引线15）为高电平时，主系统（MPU）必须使该线保持在高阻抗状态。单板之内的存储器和I/O器件可以通过跳接线安排在用户存储器的地址空间之中。
11	－12V_{DC}	－12V直流电源——系统逻辑电路用。
12	$\overline{REFREQ}$	更新请求——是为动态存储器系统单板请求总线周期的低电平有效、可“线-或”的信号。参考REFGNT（引线13）。
13	REFGNT	更新允许——是确认更新周期请求的高电平有效信号，并同时代表更新回答周期。在REFGNT为高电平（有效）时，BUSGNT保持在低电平（无效）。主系统（MPU）在REFGNT周期中是不起作用的。
14	$\overline{DEBUG}$	系统调试——表示系统中有调试硬件单板的低电平有效的静态信号。
15	BUSGNT	总线允许——是经由$\overline{BUSREQ}$信号使主系统（MPU）产生的确认请求系统总线的高电平有效信号，表示允许总线进行存取工作。在更新允许周期时，该信号无效（低电平）。
16	＋12V_{DC}	＋12V直流电源——系统逻辑电路用。

17	STANDBY	+5V直流备用电源——（早期EXORcisers系统中为+12V）。该线给电池备用存储器组件使用并和引线U相同。
18	—	保留——（在早期系统中，该线曾作为主系统（MPU）产生的时钟号、非同步信号或者对称时钟脉冲信号线，并提供给那些需要时钟本身不受处理器或存储器定时影响的外部组件使用。当使用内含时钟的器件时，就不需要再用以前那种无控制时钟信号）。
19	VXA	有效执行地址——当系统按多种存储器空间方式(又称多地址图方式)进行工作，并且程序本身正在寻址到存储器空间中的执行部分时，由主系统总线产生的高电平有效信号。另外，如果用户要求外部组件（如存储器单板）工作在执行地址空间之中，那么所有这些组件就必须能响应VXA信号。
20～22	GND	地线。
23	BS	总线状态——该信号与BA信号（引线P）一起，共同反映主系统（MPU）暂停、中断和同步状态。
24	—	保留——（早期系统为参考地）。
25～28	—	保留——这些引线及其对面引线（$\overline{C}$、$\overline{D}$、$\overline{E}$、$\overline{F}$）都被保留它用。
29	$\overline{D1}$	数据总线（第1位）——与引线$\overline{H}$上$\overline{D3}$说明相同。
30	$\overline{D5}$	数据总线（第5位）——与引线$\overline{H}$上$\overline{D3}$说明相同。
31	$\overline{D0}$	数据总线（第0位）——与引线$\overline{H}$上$\overline{D3}$说明相同。
32	$\overline{D4}$	数据总线（第4位）——与引线$\overline{H}$上$\overline{D3}$说明相同。
33	A15	地址总线（第15位）——与引线$\overline{M}$上A14的说明相同。
34	A12	地址总线（第12位）——与引线$\overline{M}$上A14的说明相同。
35	A11	地址总线（第11位）——与引线$\overline{M}$上A14的说明相同。
36	A8	地址总线（第8位）——与引线$\overline{M}$上A14的说明相同。
37	A7	地址总线（第7位）——与引线$\overline{M}$上A14的说明相同。
38	A4	地址总线（第4位）——与引线$\overline{M}$上A14的说明相同。
39	A3	地址总线（第3位）——与引线$\overline{M}$上A14的说明相同。
40	A0	地址总线（第0位）——与引线$\overline{M}$上A14的说明相同。



附图16.1 EXORboard 印制电路板标准图 (1)



附图16.2 EXORboard 印制电路板标准图（2）

附表16.1 M68MM单板系列存储器地址分配图

地址	M68MM01	M68MM01A / M68MM01A2		M68MM01B1A	M68MM01D	M68MM19	
F000–FFFF	注 1	注 2	注 2	注 1	2K U14 注 3 2K U13 注 3	4K U27 注 5	2K U28 注 4 2K U27 注 4
E000–F000	注 1	有效存储区		PIA·ACIA·PTM 256 RAM	EC20 PIA·ACIA·PTM EC10 有效存储区	PIA·ACIA·PTM 2K RAM	
D000–E000	注 1	2K U17 2K U18		Available 有效存储区	2K U12 注 7 2K U11 注 7	4K U30	2K U30 2K U29
C000–D000	1K U15 1K U14 1K U13 1K U12	2K U19 2K U20	1K U17 1K U18 1K U19 1K U20	2K U14 2K U13	2K U10 注 7 有效存储区	4K U29	2K U28 注 4 2K U27 注 4
B000–C000	有效存储区	有效存储区		有效存储区		4K U25	
A000–B000						4K U27 注 5	
8000–A000		8409 ▸ 8400 ▸ PIA1, 2·ACIA				有效存储区	
6000–8000		有效存储区					
5000–6000	580B ▸ 5800 ▸ PIA1,2,3						
2000–5000	有效存储区						
1000–2000							
0C00 / 0800 / 0400							
0000–0400	1K RAM	1K RAM		128 RAM			
		注 6		注 6	注 6	注 6	

	设置时基本地址	容量	模块（基本）
03	9FFC	4	4 (9E00) 4 (9E00)
04	A000 C000	8K 8K	8K (0000) 8K (2000)
04A	8000 C000	8K 8K	8K/16K
05A	EF00	16	16 (0000)
05B	EF00	32	32 (0000)
05C	EF00	8	8 (0000)
06	7800	2K	2K (0000)
07	EC20	8	8 (8C00)
09	0000	4K	4K (0000)
10RTC	EC40	64	64 (0000)
12	B800	2K	2K (0000)
12A	—	8	8 (0000)
13A	91FE	2	2 (0000)
13B	91FC	4	4 (0000)
13C,D	90FC	4	4 (0000)
14	EC30	4	4 (0000)
15A,A1	9D00	4	4 (0500)
15B	9D10	4	4 (0500)
15C	9D08	8	8 (0500)
DISK	E800	1K+6	—
PROM PR	EC08		—
SYST AN		528	—
MACE	EE10	8	—

注：

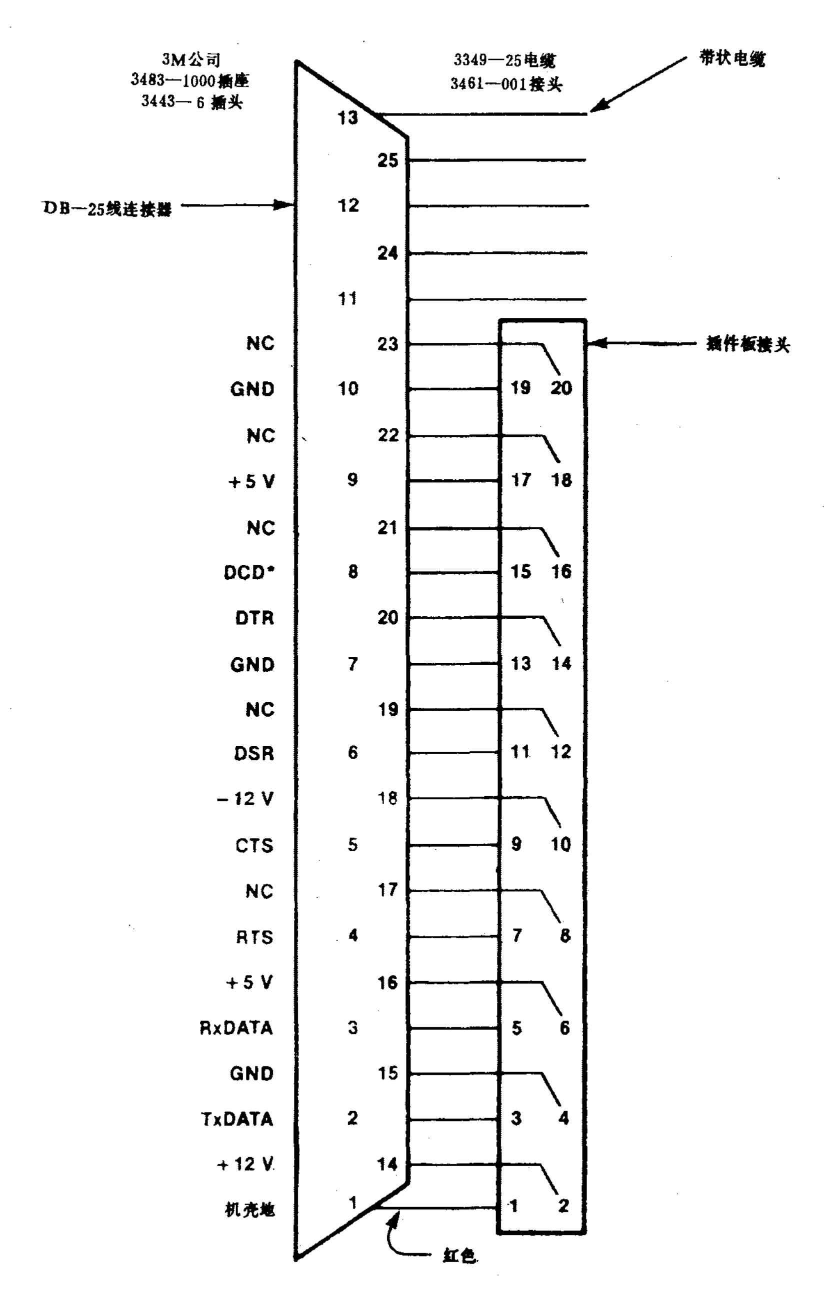
1. 可选ROM地址
2. 不用DEBUG板时，可选ROM地址
3. 用DEBUG板时，从地址图中去掉
4. 用DEBUG板时，F000～C000移到U27和U28
5. 用DEBUG板时，F000～A000移到U27
6. 用以下器件再编程时，地址图可变动
   - MM1A, A2 — 82S129 PROM
   - MM1B1A — 2ea 82S129 PROMs
   - MM1D — 82S103 FPGA
   - MM19 — 82S100 FPLA
7. 可禁止插座U10，U11，U12以扩展可用的存储器空间

▨ ——无效区

附表17 RS-232C串行接口信号

引线号	符 号	信 号 名 称 和 说 明
1	F.GND	保护接地（框架、壳体）
2	TxD	发送数据——终端发送数据给MODEM的接线
3	RxD	接收数据——MODEM给终端发送数据的接线
4	RTS	请求发送——终端请求允许给MODEM发送数据的接线
5	CTS	清除发送——MODEM确认可接收终端请求发送数据的接线
6	DSR	数据设备就绪——MODEM表示本身在线、可服务或工作状态的接线
7	GND	地线——信号和电源地
8	DCD	数据载波检测——MODEM表示其接口通信信道处于可接收的有效状态接线
9	+ 5 V	可选——接到 + 5 V_{DC}电源（通常不接）
10	GND	地线——信号和电源地
11、12、13	—	未接线
14	+ 12V	可选接到 + 12V_{DC}电源（通常不接）
15	GND	地线——信号和电源地
16	+ 5 V	可选接到 + 5 V_{DC}电源（通常不接）
17	—	未接线
18	- 12V	可选接到 - 12V_{DC}电源（通常不接）
19	—	未接线
20	DTR	数据终端就绪——终端表示本身在线、可服务或工作状态的接线
21～25	—	未接线

注：RS-232C通常用25芯接头，引线号即指25芯接线号。

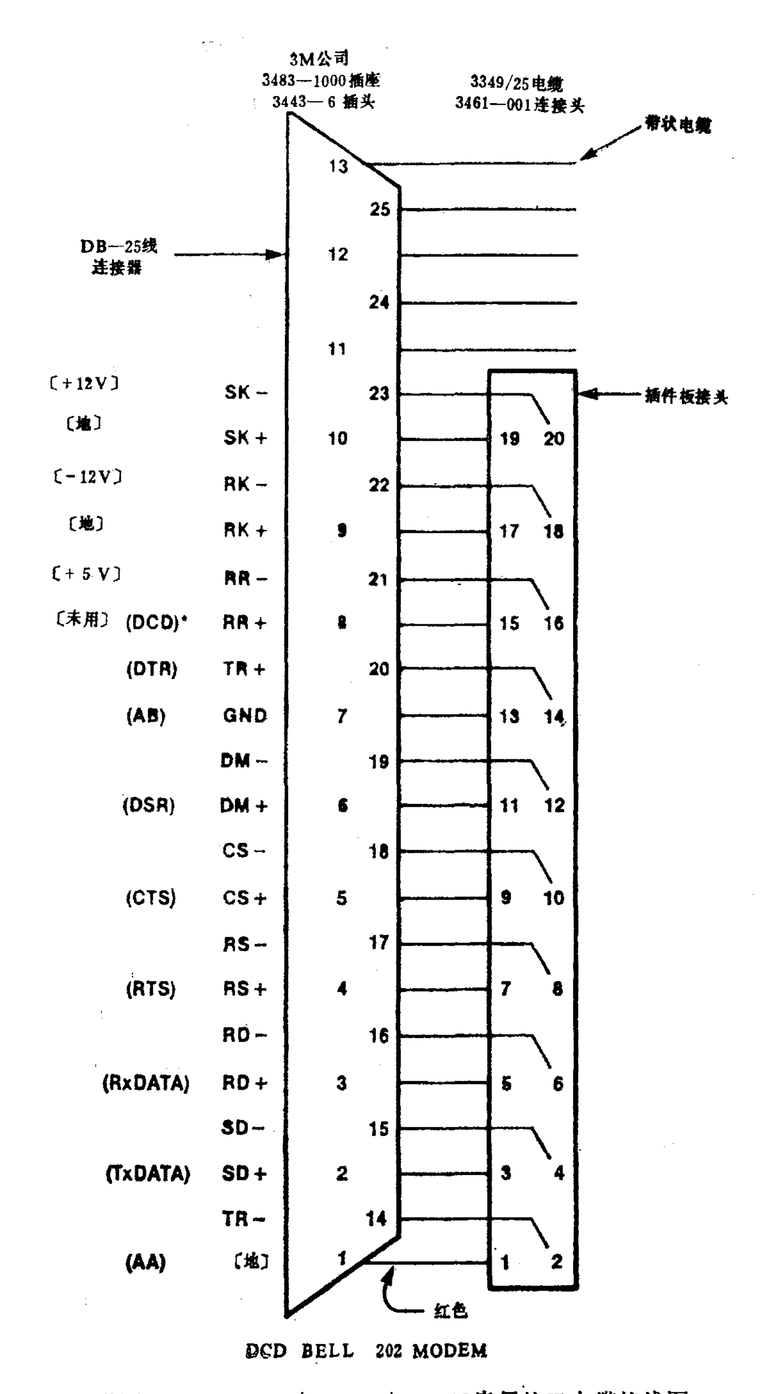


DCD是BELL 202 MODEM

附图17.1 RS-232 串行接口电缆接线图

附录18 RS-449/RS-422/RS-423串行接口信号

引线号	符　号	信号名称和说明
1	AA	机壳地——可不连
2	TxD+	发送数据——高电平有效，终端给MODEM发送数据的接线
3	RxD+	接收数据——高电平有效，MODEM给终端发送数据的接线
4	RTS+	请求发送——高电平有效，终端请求允许给MODEM发送数据的接线
5	CTS+	清除发送——高电平有效，MODEM确认可接收终端请求发送数据的接线
6	DSR+	数据设备就绪——高电平有效，MODEM表示本身在线、可服务或工作状态的接线
7	GND	地线——信号地
8	DCD+	数据载波检测——高电平有效，MODEM表示其接口通信信道处于可接收的有效状态的接线
9	RDCLK+	接收时钟——高电平有效，进入终端的时钟信号以便使数据同步的接线
10	TDCLK+	发送时钟——高电平有效，进入MODEM的时钟信号以便使数据同步的接线
11～13	—	未接线
14	DTR-	数据终端就绪——低电平有效（可选接到+12V_{DC}电源）
15	TxD-	发送数据——低电平有效
16	RxD-	接收数据——低电平有效（可选接+5V_{DC}电源）
17	RTS-	请求发送——低电平有效
18	CTS-	清除发送——低电平有效（可选接-12V_{DC}电源
19	DSR-	数据设备就绪——低电平有效
20	DTR-	数据终端就绪——高电平有效，终端表示本身在线、可服务或工作状态的接线
21	DCD-	数据载波检测——低电平有效
22	RDCLK-	接收时钟——低电平有效
23	TDCLK-	发送时钟——低电平有效
24～25	—	未接线



附图18.1 RS-449/RS-442/RS-423串行接口电缆接线图

附录19 6809习题问答

1. 习题

（1）6800和6809指令系统之间在什么级别上兼容？

（2）6809指令系统有多少条基本指令？

（3）6809可使用多少种不同的寻址方式？

（4）6809主要是为____市场设计的。

（5）6809设计的两个首要考虑是——和——。

（6）在MOTOROLA 8 位微处理器系列中最先进的微处器是——。

（7）MOTOROLA 8 位微型计算机系列中最先进的微型机是——。

（8）6809和6809E之间的根本区别是什么？

（9）6809比6800增加了哪些寄存器？

（10）在CC寄存器中两个新的条件码标志和它们的位置是——和——。

（11）6809内部寄存器中有哪些可以变址并能作为变址寻址方式使用？

（12）当一条指令指出一个操作数地址的地址，该寻址方式叫——寻址方式。

（13）6809的分支转移范围多大？和6809相比，6800有多大范围？

（14）6809的直接寻址方式和6800相比有何不同？

（15）3.2MHz石英晶体可产生——MHz的6809的时钟？

（16）6809响应快速中断请求（$\overline{FIRQ}$）时，只有——和——寄存器放入堆栈。

（17）相移时钟信号Q和工作时钟信号E频率相同，但Q超前于E多少度？

（18）当6809上没出现有效存储器地址时，地址总线上是什么状态？

（19）6809用——取代了6800的三态控制线（TSC）。

（20）在6809上没有的，但在6809E上有的两条外部控制线是——和——。

（21）固有寻址也称为——寻址。

（22）包括内部寄存器间数据传送或交换的指令用——寻址。

（23）说明6809的直接寻址和6800的直接寻址之间有何区别？

（24）DPR作为直接寻址的——字节并作为一部分指令的——字节。

（25）6809用的二种相对寻址是——和——。

（26）6809用的二种分支转移是——和——。

（27）说明6809长分支转移指令格式。

（28）在地址2000中的分支转移指令相对地址偏移是F150，决定分支转移目的地址为何？

（29）说明两种PC相对寻址？

（30）写出具有变址寻址方式的寄存器名称？

（31）变址寻址的四种基本形式是——、——、——和——。

（32）具有变址寻址指令的操作码总跟有——。

（33）常数偏值变址寻址使用三种常数偏值是——、——和——。

(34) 为什么说5位有符号的偏值方式是最有效的常数偏值的变址寻址方式?

(35) 16位带符号偏值的范围从——到——。

(36) 说明累加器偏值变址寻址比常数偏移变址寻址的优点?

(37) 自动加/减的另一名称可以叫作——。

(38) 使用自动加/减方式规定的指示器(寄存器)可以自动加或减——或——。

(39) 说明作成变址寻址方式后缀字节的数字位规定的范围和它对应的数字位的安排。

(40) 当5位带符号偏值不包含在后缀字节时,5位偏值范围(第7位)必须是——。

(41) 如果要把用常数偏值为10_{16}的Y变址寄存器规定的存储器单元的内容装入累加器A,那么应用什么汇编代码表示这种操作?

(42) 为了实现(41)题的操作需要什么后缀字节?

(43) 已知变址寻址的后缀字节为83_{16},请解释其含意并决定适当的汇编代码?

(44) 定义间接寻址?

(45) 6809寻址方式中有哪些可以用间接寻址?

(46) 为了表示间接寻址所用的汇编语言符号是——。

(47) 已知下列一段程序,装入X变址寄存器中的内容是什么?

```
 ⋮
04FB    LDA #
04FC    10
04FD    LDU #
04FE    FC
04FF    50
0500    LDX [A, U]
0501    POST byte
 ⋮
06FE    C7
06FF    F5
0700    1B
0701    AA
 ⋮
FC5E    D5
FC5F    C7
FC60    06
FC61    FF
FC62    00
 ⋮
```

(48) 在上述操作中需要什么样后缀字节(POST-byte)?

(49) 使用扩充间接寻址的指令需要多少字节?

(50) EXG指令是交换两个等长的寄存器内容,用下列指令时哪两个寄存器被交换?

EXG

8 B

(51) 列出6809的8位寄存器？

(52) 列出6809的16位寄存器？

(53) 哪些16位的寄存器可以用作变址寄存器？

(54) 在子程序和中断时，6809中有哪些寄存器自动地使数据进栈？

(55) X和Y寄存器有什么区别？

(56) Z标志位何时被置1？

(57) 在运算结果中哪个CCR标志位表示符号错？

(58) 6809在响应快速中断请求时，F标志位必须被——。

(59) 在FIRQ时有哪些寄存器要进栈？

(60) E标志位的功能是什么？

(61) 6809中有哪些内部寄存器可以被装入数据或输出存储数据？

(62) 解释说明下列两程序间的区别？

```
LDX #          LDX $ $
FC             FC
50             50
```

(63) EA→Y是——指令用的操作符号。

(64) 写出从S寄存器内容中减7的一条指令。

(65) 写出把累加器B的内容加到Y寄存器的指令。

(66) 写出把累加器A的内容加到U寄存器，并把结果送到X寄存器的指令。

(67) 6809有哪些寄存器可以作为堆栈指示器使用？

(68) 6809堆栈指示器总指向——（哪里？）

(69) 怎样把X和Y寄存器用作堆栈指示器。

(70) 用X作为进栈顺序的指示器，编写出使Y寄存器、累加器A和条件码寄存器进栈的指令顺序。

(71) 根据前一问题，为什么说下面的回答不正确？

```
STY  , ——X
TFR  CC, B
STD  , ——X
```

(72) 按（70）问之中所设堆栈，写出出栈指令顺序。

(73) 当使用PSH指令时，进栈的顺序是什么？

(74) 按U堆栈指示器写出累加器A和B、X寄存器和程序计数器进栈的顺序。

(75) 按上一问题的指令写出正确的操作码清单？

(76) ABX和LEAX B, X两者有何差别？

(77) 为什么MUL操作是一种无符号的乘法操作？

(78) 哪种6809算术指令可把累加器B中的带符号的8位数转换为在累加器D中带符号的16位数？

(79) 写一段指令程序，其条件如下：用累加器A和B之积算出U的堆栈指示器内容，然后用该指示器规定的U堆栈寄存器内容把全部内部寄存器放入堆栈之中。

(80) 按问题 (79) 的指令写出正确的操作码。

(81) 在问题 (79) 和 (80) 中执行程序时需要多少MPU周期？

(82) 算术左移ASL和逻辑左移LSL二者有何差别？

(83) ANDCC和ORCC指令有何作用？

(84) 怎样使CCR的标志位清零？

(85) 怎样使CCR的标志位置 1 ？

(86) 说明下述操作意义？

A. ORCC # FF　　B. ANDCC # AF　　C. ORCC # OO

(87) 三种主要测试指令是什么？

(88) 测试指令的用途是什么？

(89) BIT指令执行——操作，只对——和——CCR标志位起作用。

(90) CMP指令执行——操作。

(91) 6809有哪些寄存器可以进行比较？

(92) CMP指令影响CCR哪些标志位？

(93) TST指令可测试什么？测试表示什么意思？

(94) TST指令影响CCR哪些标志位？

(95) 执行下述指令后，判断CCR标志位的状态（设累加器A为 5 F）：

A. BITA # 20　　B. CMPA 5 F　　C. TSTA

(96) 6809三个无条件分支转移指令是——、——和——。

(97) BSR和JSR有什么相似处？又有何不同？

(98) 条件分支转移可以分为三类，它们是——、——和——条件分支转移。

(99) BLO可以用——代替，因这两种分支转移的转移测试相同。

(100) 与BMI分支转移条件相反的是——；与BGE对应的是——；与 BHI 对应的是——。

(101) 相对寻址的主要优点是——。

(102) 下面两条指令之间功能上有何差别？

BRA 05　　JMP 3, PC

(103) 在执行CWAI时，为什么条件码寄存器的E位置 1 ？

(104) 在CWAI期间可以响应中断请求（IRQ），需使I位置 0 ，而所有其它标志不变，那么CWAI的汇编指令是什么？

(105) 6800用什么汇编指令可代替上述CWAI指令？

(106) 有一程序如下，问何处读出向量地址？

```
    ⋮
LDA #10          装入控制字节
LDX #0010        向量表起始地址
```

```
    CLRB
    LBRN
  ┌→ADDB 02
  │ LSRA
  └─BCC  FB ──┐         如进位为1，去向量表
    JMP [B,X] ←┘
       ⋮
```

(107) 有哪些软件中断自动地置‘1’F和I位来屏蔽FIRQ和IRQ中断？

(108) 6809表示中断已接收的是——线。

(109) SWI 2 中断向量地址是——。

(110) 有时和向量有关的寻址方式是——。

(111) 何种指令可以代替以下指令：

```
PSHS    ALL
JMP     [FFF4]
```

(112) 为何在上问中指令不严格是SWI 2？

(113) 何种指令可以用PULS PC来代替？

(114) 何种指令可以代替RTI？

(115) SYNC指令的主要目的是什么？

(116) 当处于同步（SYNCing）状态时，如果接收了FIRQ，而且条件码寄存器的F标志位被置1，那么这时要发生什么情况？

(117) 在前一问题中如果F标志位被置0，又将发生什么情况？

(118) SYNC与CWAI有何区别？

(119) 6809等效于6800的ABA指令是什么指令？

(120) 6809等效于6800的DEX指令是什么指令？

(121) 3.2MHz的晶体将提供6809内部时钟频率为——。

(122) 驱动6809用外部TTL或CMOS时钟信号时，需怎样连接XTAL和EXTAL引线？

(123) 整个MPU速度的最好量度是——。

(124) 6809或6809E数据和地址线的驱动能力怎样？

(125) 三条6809总线状态线是——、——和——。

(126) 当MPU地址和数据线都处在高阻抗状态时，总线有效线（BA）将处在——电平。

(127) BA和BS状态线所表示的四种MPU状态是——、——、——和——。

(128) 设BA = 0、BS = 1和$A_3A_2A_1 = 110$，问MPU在做何工作？

(129) 使用74154译码器，BS经反向器接其G 2端，BA接其G 1和A端，A_1、A_2、A_3分别接其B、C、D端，在问题（127）所给条件下74154哪条输出线工作？

(130) 当在$\overline{DMAREQ}$端加低电平（逻辑0）时，总线状态线将表示MPU处于哪些状态？

(131) 6809怎样表示一个无效存储器地址？

(132) 怎样才是无效存储器地址状态，与$\overline{RESET}$中断向量取出有何区别？

(133) E_{out} (E_{in}) 和Q_{out} (Q_{in}) 之间是什么关系？

(134) 6809/6809E数据总线上何时数据有效？

(135) 6809MREADY端的功能是什么？

(136) 如果低电平加到$\overline{HALT}$端时，出现了非屏蔽中断$\overline{NMI}$将发生什么情况？

(137) 如果$\overline{HALT}$端处在低电平时，当$\overline{DMAREQ}$端加上低电平后将发生什么情况？

(138) 控制DMA功能的器件称为——，在6800系列中表示为MC——。

(139) 用6809来实现直接存储器存取的三种方法是——、——和——。

(140) 用DMAREQ线实现周期窃取DMA时，为了更新内部寄存器的数据内容，MPU将以每次——MPU周期来窃取地址/数据总线。

(141) 6809/6809E硬件中断是边沿触发的只有——中断。

(142) 当系统加电时，$\overline{RESET}$保持为低电平的时间有多长？

(143) $\overline{RESET}$端为高电平（逻辑 1）时为——伏？

(144) 可中断RESET时序的只有——和——两种事件。

(145) 什么事件可以中断NMI序列？

(146) $\overline{IRQ}$和$\overline{FIRQ}$中断时序之间有何差别？

(147) $\overline{IRQ}$或$\overline{FIRQ}$哪个中断优先级高？

(148) 6809E最好是用到——系统中。

(149) 在6809E中增加了三种总线状态线，它们是——、——和——。

(150) 6809E中——线和6809$\overline{DMAREQ}$线相似。

2. 习题答案

(1) 源码级兼容（记忆符Mnemonic)

(2) 59

(3) 19

(4) 系统

(5) 位置独立程序和6800兼容

(6) 6809

(7) 6801

(8) 6809E是芯片上无时钟型的6809

(9) 累加器D (D)、直接页面寄存器 (DP)、Y变址寄存器、用户堆栈指示器

(10) F（6 位）、E（7 位）

(11) 全部变址寄存器和堆栈指示器（X、Y、S 和 U)

(12) 间接寻址

(13) 6809可以用长相对寻址方式分支转移到64K存储器空间任何地址上，而6800分支转移范围只限于$+127_{10}\sim-128_{10}$

(14) 在直接寻址方式中6809用直接页面寄存器 (DP) 可以访问64K存储器中的任何地址，而6800仅限于存储器中前256字节（页面 0）

(15) 0.8MHz

(16) 程序计数器 (PC) 和条件码 (CC) 寄存器

(17) 相移90^{0}

(18) 当没有总线安排时，在任何时钟信号期间6809地址是$FFFF_{16}$

(19) 直接存储器访问请求（$\overline{DMAREQ}$）

(20) 工作忙和最后一条指令周期（LIC）

(21) 如果操作有一个累加器，则称隐含或累加器寻址

(22) 寄存器

(23) 6800的直接寻址被限制在前256字节的存储器。6809的直接寻址，使用直接页面寄存器（DPR）可以使用全部64K字节存储器地址。

(24) 最高位、最低位

(25) 分支转移相对寻址和程序计数器相对寻址

(26) 短分支转移和长分支转移

(27) 2字节指令的操作码后跟2字节相对地址偏移（除去LBRA和LBSR，它们的操作码只有一个字节）

(28) 因为偏移量是2字节（F150），需要长分支转移指令。长分支转移指令用4字节；所以程序计数器内容是2000＋4＝2004，现在偏值的最高位是1，所以6809将向回转移。为了得到目的地址，要把2的补码偏值加到程序计数器之中。所以目的地址是2004＋F150＝1154

(29) 带符号的8位程序计数器相对寻址的偏移和带符号的16位程序计数器相对寻址偏移

(30) X、Y、S和U

(31) 零偏移、常数偏移、累加器偏移和自动加/减变址

(32) 后缀字节

(33) 5、8和16位带符号的偏值

(34) 因为5位带符号偏值算做后缀字节的一部分

(35) －32768到＋32767（±15位）

(36) 用累加器偏值，在变址操之前，该偏值可进行计算

(37) 后增量/前减量

(38) 1或2

(39) 寻址方式范围（0～3位），间接范围（4位），指示寄存器范围（5和6位），5位偏值范围（7位）

(40) 置1（逻辑1）

(41) LDA　10，Y

(42) $10101000_2 = A8_{16}$

(43) 该后缀字节为X变址寄存器被自动减2。在每执行一条用该后缀字节的指令之后，X变址寄存器则被减2。汇编代码是 ,－－X

(44) 在间接寻址条件下，被寻址的存储器单元指的是操作数的地址，而不是操作数本身

(45) 扩充寻址、程序计数器相对寻址和所有变址寻址方式，除去自动加/减之外

(46) 方括号〔 〕

(47) 在LDX指令之前，累加器A被装入10，指示寄存器U被装入FC50。LDX指令用

的是对指示寄存器U，偏值为累加器A内容的间接寻址。所以操作数地址所在的存储器单元是0010+FC50=FC60。从程序表格中可在这个地址中找到内容06为操作数地址的高位字节。低位字节在地址FC61之中。这样操作数所在地址为06FF。在该地址中的内容是F5。但X变址寄存器是16位的寄存器。所以F5要装入的是高位字节，而下一个相邻存储器单元的内容将为低位字节。所以装入变址寄存器中的内容是F51B

(48) $11010110_2 = D6_{16}$

(49) 4或5；指令操作码字节（S），跟一后缀字节，跟一2字节地址

(50) 累加器A和直接页面寄存器

(51) 累加器A（A）、累加器B（B）、直接页面寄存器（DPR）、条件码寄存器（CCR）

(52) 累加器D（D）、程序计数器（PC）、X寄存器（X）、Y寄存器（Y）、U寄存器（U）、S寄存器（S）

(53) X、Y、S或U寄存器

(54) S寄存器

(55) 没有区别。完全用同样的方法使用它们，也完全按同样的指令系统进行操作。

(56) 当运算结果为0时

(57) V标志位

(58) 清零（逻辑0）

(59) 只有程序计数器和条件码寄存器

(60) E标志位同F标志位一起使用。当响应快速中断请求时，E标志位自动地被清零，而在其它中断时，被置1。6809用该标志位来判断需要出栈的个数

(61) 累加器A、B和D，以及X、Y、S和U寄存器

(62) LDX #指的是X寄存器装入的为立即数FC50；而LDX $ $指的是X的高位节装入的是FC50中的内容，低位字节装入的是FC51中的内容

(63) 把有效地址装入Y中（LEAY）

(64) LEAS -7，S

(65) LEAY B，Y

(66) LEAX A，U

(67) X、Y、S和U寄存器

(68) 指堆栈顶部或者把最后的数值压入堆栈之处

(69) 使用X和Y寄存器的自动加或自动减的变址寻址方式，并与各种装入和存储指令一起使用

(70)
```
STY  ,--X
STA  ,-X
TFR  CC, B
STB  ,-X
```

(71) 该指令顺序是使Y寄存器、CCR条件码寄存器进栈，但没有按（70）问之中规定累加器A的进栈顺序

(72) LDB ,X+ 注意数据出栈时必须按进栈顺序相反方向出栈

TFR B, CC
LDA , X+
LDY , X++

(73) CC、A、B、DP、X、Y、U/S、PC，存储器加1方向

(74) PSHU A, B, X, PC

(75) PSHU操作码：36_{16}

后缀字节：96_{16}

(76) ABX是一种固有指令，是把累加器B的无符号的内容加到X寄存器中，LEAX B, X是2字节指令，是把带符号的累加器B中的内容加到X寄存器中

(77) 为了实现多精度（多字节）的乘法操作

(78) 带符号的扩展指令（SEX）

(79) MUL

TFR D, U

PSH U CC, A, B, DP, X, Y, S, PC

(80) 正确的操作码清单是：

```
MUL        3D
TFR D, U   1F
           03
PSHU       36
           FF
```

(81) 上述程序需要35个MPU执行周期。MUL指令需要11周期，TFR需要7周期，PSHU需要17周期。注意PSHU指令每进栈一个字节需要5个时钟周期加1个周期

(82) 这两条指令之间没有功能上差别

(83) ANDCC和ORCC指令可以置‘1’和置‘0’到CCR标志位

(84) 为清零CCR标志位用ANDCC指令分别与标志位所在位置同逻辑0相‘与’

(85) 为置‘1’CCR标志位用ORCC指令分别与标志位所在位置同逻辑‘1’相‘或’

(86) A. 该指令将使CCR所有各位置1，因为逻辑1与CCR所有各位相‘或’

B. 该指令将使F和I位置0，因为逻辑0与CCR的第4和第6位相‘与’

C. 该指令没有任何作用，因为逻辑0与CCR所有各位相‘或’

(87) 逻辑位测试（BIT）、算术比较测试（CMP）、字节测试（TST）指令，测试0、正或负值

(88) 为测试只影响条件码寄存器的数据，以便进行条件分支转移。其数据本身结果不变

(89) ‘与’，N、Z

(90) 减法

(91) 累加器A、B、和D任何一个可变址的寄存器（X、Y、S、U）

(92) N、Z、V和C标志位

(93) 测试指令可以对累加器A和B或存储器单元进行正、负或零数值的测试

(94) 只影响N和Z标志位

(95) A. N位置‘0’，Z位置1表示与操作结果为零，其它所有各位不受影响和前次的操作结果相同可能是1或0

B. Z位置1，表示减法操作结果为0。C位也被置1，因为2的补码操作结果最后产生进位（或借位）。N和V位被置0，其它所有各位不受操作影响，和前次操作结果一样可能是1或0

C. N位被置0，表示累加器A为正值。Z位也为0表示累加器A为非零值。在TST操作时V位总为0，所有其它位不受操作影响，和前次操作结果一样可能是1或0

(96) 无条件分支转移（BRA）、分支转移子程序（BSR）和非分支转移（BRN）

(97) 相似处是：它们都用来调用子程序，并使程序计数器内容保留在S堆栈之中。另外它们都需用RTS作最后一条子程序指令以便返回主程序

它们区别是：BSR使用相对寻址，而JSR可用直接、扩充或变址寻址

(98) 简单、带符号、无符号条件分支转移

(99) BCS

(100) BPL、BLT、BLS

(101) 位置独立（程序浮动）

(102) 功能上没有差别。JMP指令将使程序分支转移到BRA指令相同的单元。注意，JMP是用程序计数器相对寻址，所以可位置独立

(103) 在执行CWAI指令期间条件码寄存器E位置1，是因为在该操作时，所有内部寄存器（除S外）都要保留在堆栈之中

(104) CWAI # EF

(105) CLI

WAI

(106) 001E

(107) SWI1

(108) 是总线状态（BS）输出线

(109) FFF4:FFF5

(110) 绝对间接寻址

(111) SWI2

(112) 进栈时E标志位不需要置1。为了置‘1’E标志位，在PSHS指令之前应增加ORCC#80指令

(113) RTS

(114) PULS ALL

(115) 为了用外部硬件事件来同步主程序，如数据传送

(116) 同步状态将被清除，下一条指令将被执行

(117) 同步状态将被清除，将要执行FIRQ中断服务程序

(118) SYNC不象CWAI那样使6809各寄存器进栈，另外，CWAI不象SYNC那样允许外部硬件同步

(119) PSHS B

ADDA ,S+

(120) LEAX —1, X

(121) 0.8MHz

(122) TTL或CMOS的信号加到EXTAL，而XTAL接到地上。所加频率必须是所要求频率的4倍数值

(123) 处理器的处理量（吞吐量）

(124) 一个标准的肖特基TTL负载加上8个在额定总线速度下的6800系列器件

(125) 总线有效（BA）、总线状态（BS）和读/写（$R/\overline{W}$）

(126) 高

(127) 正常状态、中断响应（IACK）状态、同步响应状态和暂停/总线回答状态

(128) BA＝0和BS＝1说明在中断响应（IACK）MPU状态，并表示在取出中断向量。因为A_3、A_2、$A_1=110_2$，则低四位地址线（A_3、A_2、A_1、A_0）将为1100_2（C_{16}）或1101_2（D_6）。在此情况下非屏蔽中断（$\overline{NMI}$）向量进行取出。注意在译码中断向量时，A_{10}的状态无须知道。

(129) 第12线

(130) 暂停/总线允许

(131) 地址线A_0～A_{15}均为高电平（$FFFF_{16}$），$R/\overline{W}=1$和BS＝0

(132) BS＝0为无效存储器地址；BS＝1为$\overline{RESET}$中断向量取出

(133) E_{out}（E_{in}）和6800Φ2时钟相似。Q_{out}（Q_{in}）叫相移90^0的正交时钟信号。E_{out}（E_{in}）和Q_{out}（Q_{in}）频率完全相同，然而Q_{out}（Q_{in}）超前于E_{out}（E_{in}）$90°$或1/4时钟周期

(134) 当E_{out}（E_{in}）处在逻辑1电平（高电平）时

(135) 为了延长E_{out}和Q_{out}脉冲最大到10μs，这样慢速外部设备就有时间去响应6809的信号

(136) $\overline{HALT}$状态不会被中断。但$\overline{NMI}$的工作状态将被锁存下来，当$\overline{HALT}$端返回高电平时，就会执行$\overline{NMI}$序列

(137) 将要执行$\overline{DMAREQ}$事件序列

(138) 直接存储器存取控制器（DMAC），MC6844

(139) 暂停方式（$\overline{HALT}$端），周期窃取（$\overline{DMAREQ}$端）和总线多路转接方式（外部逻辑）

(140) 第15个

(141) 非屏蔽中断$\overline{NMI}$

(142) 保持到内部时钟振荡器完全工作，接于20ms

(143) 最小4V

(144) $\overline{HALT}$和$\overline{DMAREQ}$

(145) 下列之一的情况：另一个$\overline{NMI}$，$\overline{RESET}$，$\overline{DMAREQ}$，$\overline{HALT}$

(146) A. $\overline{IRQ}$置‘1’E标志位；$\overline{FIRQ}$置‘0’E标志位

B. $\overline{IRQ}$使全部内部寄存器进栈；$\overline{FIRQ}$只使PC和CCR进栈

C. $\overline{IRQ}$置‘1’I标志位，$\overline{FIRQ}$置‘1’F和I标志位

D. $\overline{IRQ}$向量地址在FFF8:FFF9

$\overline{FIRQ}$向量地址在FFF 6：FFF 7

(147) $\overline{FIRQ}$

(148) 多处理器

(149) BUSY、LIC和AVMA

(150) 三态控制线TSC

附录20　6809操作码表

附表20.1　6809操作码表

低四位	高四位 DIR 0000 0	 0001 1	 REL 0010 2	 0011 3	 ACCA 0100 4	 ACCB 0101 5	 IND 0110 6	 EXT 0111 7	 IMM 1000 8	 DIR 1001 9	 IND 1010 A	 EXT 1011 B	 IMM 1100 C	 DIR 1101 D	 IND 1110 E	 EXT 1111 F	
0000 0	6 NEG	PAGE2	3 BRA	4+1 LEAX	2	2 NEG	6+1	7	2	4 SUBA	4+1	5	2	4 SUBB	4+1	5	0
0001 1	—	PAGE3	3 BRN/ 5 LBRN	4+1 LEAY	—	—	—	—	2	4 CMPA	4+1	5	2	4 CMPB	4+1	5	1
0010 2	—	2 NOP	3 BHI/ 5(6) LBHI	4+1 LEAS	—	—	—	—	2	4 SBCA	4+1	5	2	4 SBCB	4+1	5	2
0011 3	6 COM	2 SYNC	3 BLS/ 5(6) LBLS	4+1 LEAU	2	2 COM	6+1	7	4,6,6+1,7 SUBD	5,7,7+1,8 CMPD		5,7,7+1,8 CMPU	4	6 ADDD	6+1	7	3
0100 4	6 LSR	—	3 BHS 5(6) (BCC)	5+1/by PSHS	2	2 LSR	6+1	7	2	4 ANDA	4+1	5	2	4 ANDB	4+1	5	4
0101 5	—	—	3 BLO 5(6) (BCS)	5+1/by PULS	—	—	—	—	2	4 BITA	4+1	5	2	4 BITB	4+1	5	5
0110 6	6 ROR	5 LBRA	3 BNE/ 5(6) LBNE	5+1/by PSHU	2	2 ROR	6+1	7	2	4 LDA	4+1	5	2	4 LDB	4+1	5	6
0111 7	6 ASR	9 LBSR	3 BEQ/ 5(6) LBEQ	5+1/by PULU	2	2 ASR	6+1	7	—	4	4+1 STA	5	—	4	4+1 STB	5	7
1000 8	6 ASL (LSL)	—	3 BVC/ 5(6) LBVC	—	2	2 ASL (LSL)	6+1	7	2	4 EORA	4+1	5	2	4 EORB	4+1	5	8
1001 9	6 ROL	2 DAA	3 BVS/ 5(6) LBVS	5 RTS	2	2 ROL	6+1	7	2	4 ADCA	4+1	5	2	4 ADCB	4+1	5	9
1010 A	6 DEC	3 ORCC	3 BPL/ 5(6) LBPL	3 ABX	2	2 DEC	6+1	7	2	4 ORA	4+1	5	2	4 ORB	4+1	5	A
1011 B	—	—	3 BMI/ 5(6) LBMI	6/15 RTI	—	—	—	—	2	4 ADDA	4+1	5	2	4 ADDB	4+1	5	B
1100 C	6 INC	3 ANDCC	3 BGE/ 5(6) LBGE	20 CWAI	2	2 INC	6+1	7	4,6,6+1,7 CMPX	5,7,7+1,8 CMPY		5,7,7+1,8 CMPS	3	5 LDD	5+1	6	C
1101 D	6 TST	2 SEX	3 BLT/ 5(6) LBLT	11 MUL	2	2 TST	6+1	7	7 BSR	7	7+1 JSR	8	—	5	5+1 STD	6	D
1110 E	3 JMP	8 EXG	3 BGT/ 5(6) LBGT	—	—	—	3+1 JMP	4	3,5,5+1,6 LDX		4,6,6+1,7 LDY		3,5,5+1,6 LDU		4,6,6+1,7 LDS		E
1111 F	6 CLR	7 TFR	3 BLE/ 5(6) LBLE	19/20/20 SWI/2/3	2	2 CLR	6+1	7	—	5,5+1,6 STX		6,6+1,7 STY	—	5,5+1,6 STU		6,6+1,7 STS	F

参考文献资料

〔1〕 微型计算机基础技术手册
〔日〕横井与次郎著　刘德贵译
科学出版社出版　81年6月

〔2〕 Multiprocessing With Motorola's MC6809E Hunter Scales July 1981 BYTE P.136

〔3〕 MEK6809D4 AND MEK68KPD USER'S MANUAL Memory Systems for MOTOROLA INC.

〔4〕 MOTOROLA Microprocessors DATA MANUAL MOTOROLA INC., 1981

〔5〕 MC6809-MC6809E Microprocessor Programming Manual MOTOROLA INC., 1981

〔6〕 6809 Microcomputer Programming & Interfacing, With Experiments Andrew C. Staugaard, Jr. Howard W. Sams & Co., Inc. 1981

〔7〕 Programming Microprocessor Interfaces for Control and Instrumentation Michael Andrews Prentice-Hall Inc. 1982

〔8〕 6809 Assembly Language Programming Lance Leventhal McGraw-Hill Inc. 1981

〔9〕 6809 ハンドブック
加瀬　清著　アスキー出版　1982

〔10〕 APPLICATION NOTE FOR 6809 (MOTOROLA公司)

(1) AN-820 Hardware Considerations for Direct Memory Access Using the MC6809 Microprocessor Unit and MC6844 DMA Controller
(MC6809微处理器和MC6844DMA控制器作DMA的硬件考虑)
本文献讨论DMA设计，其中考虑处理器和DMA控制器之间总线控制变换过程中所要求的无效周期的保护问题。接口设计完全满足标准的时间关系要求和总线保护要求。

(2) AN-825 An Interactive Graphic System Using the MC6809
(应用MC6809实现交互图形系统)
本文讨论使用廉价视频图形发生器（VDG）——MOTOROLA MC6847的微处理器系统的图形处理能力，低成本显示系统，MC1372彩色TV视频调制器，MC6809微处理器智能控制器。给出系统的完整的方案。

(3) AN-830 An Intelligent Terminal With Data Link Capability (MC6809, MC6854, MC6845, MC6850, MC6844)（数据通信智能终端）
本文献讨论设计体积小功能强的数据通信终端的方法，使用高速数据通信链络器件构成MC6809为核心的终端处理机。文中给出了详细的硬件电路和全部的通信控制程序的流程图和源程序清单。使用这些程序可以进一步做出操作系统，

实际构成文字处理机、销售终端机或数据输入源设备等。因为具有数据通信能力，可以调用远程计算机资源，同步串行传输速率可达1.5MHz。

（4）AN-831 An IEEE-488 Bus Interface Using DMA（DMA式IEEE-488总线接口）

本文献介绍使用MC6809处理器构成收/发的IEEE-488系统。文中综述数据传送操作、通用接口总线（GPIB）和某些DMA技术，同时对以前做出的实际系统进行了评价。

（5）AN-833 Software Refresh Memory Board for an MC6809 System
（MC6809系统使用软件更新存储器板）

本文献说明在任何MC6809系统中，使用软件更新的动态存储器板时的硬件和软件的要求。软件更新的方法仅占用处理机5%的时间，而且所用硬件也较少，同时还可使用慢速RAM，比其它硬件方法优越。文中所设计的线路采用MCM4116 16K RAM，但也可以使用MCM4027 4KRAM和MCM6664 64KRAM。

（6）AN-835 64K Dynamic RAM Memory Board With Transparent Refresh
（透明更新式64K动态存储器板）

本文献介绍在6800或6809系统中，使用硬件更新方法的64K动态存储器板，并给出了完整的逻辑框图。

（7）AN-836 Using Low-cost 1 MHz Peripherals in a 2 MHz System With the MC68B09 and MC68B09E
（MC68B09和MC68B09E 2 MHz系统使用低成本1 MHz外围系统）

本文献说明在2 MHz的MC68B09和MC68B09E系统中，如何使用慢速外围器件和存储器的设计方法。通常速度提高后，系统成本就会增加。文中介绍具体使用MC68B09和MC68B09E实现高速微处理器系统怎样配合慢速系统工作。

（8）AN-839 A Data Communications System Using an MC6809 MPU, MC-68652MPCC, and/or the MC68661 EPCI
（MC6809（MPU）、MC68652（MPCC）和MC68661（EPCI）组成数据通信系统）

本文献介绍使用MC6809和MC68652及MC68661共同构成数据通信系统，可以减少系统复杂性，提高系统性能。文中着重介绍了使用MC68661高级程控通信接口（EPCI）和MC68652多协议通信控制器（MPCC）同MC6809组成系统的硬件设计方法，同时给出了6809指令系统测试这两种器件的软件程序。

（9）AN-850 Multi-Processor Controller Using the MC6809E and the MC-68120
（MC6809E和MC68120构成多处理器控制器）

本文献说明当要求提高系统性能时，设计人员的主要任务是怎样提高系统的吞吐能力。为了得到高性能、灵活方便的系统，怎样在当前系统中增加扩展能力，而不对整个系统重新设计，文中为设计人员提供了两种方法：即使用单处理器和多处理器的方法。研究讨论了这两种方案，并重点说明了使用MC6809E和MC68120的多处理器系统的结构和设计原理。

〔11〕 MC6809专集
《微型机与应用》 1982年第3、4期合刊
〔12〕 SELECTED PAPERS ON M6809 MICROPROCESSOR SYSTEMS M6809
マイクロプロセッサーミステム論文選
刘德贵选编 1982年内部交流（英3-5/3495）
〔13〕 MC6809 PRELIMINARY PROGRAMMING MANUAL MOTOROLA INC., 1979
〔14〕 Advance Information MC6809/MC68A09/MC68B09 MOTOROLA INC., 4/1980, 1983
〔15〕 Advance Information MC6809E/MC68A09E/MC68B09E MOTORLA INC., 11/1980, 1983
〔16〕 Advance Information MC6829/MC68A29/MC68B29 MOTOROLA INC., 8/1980, 1983
〔17〕 Fast Fourier for the 6800,
Richard H. Lord, BYTE Vol.4, NO.2 PP,108～119, Feb, 1979
〔18〕 8080A マイクロプロセッサを用いた高速フーリエ変換処理プログラム
世古，成田，《インターフェース》1978年8月号PP.41～50
〔19〕 高速フーリエ変換によるパワー・スペクトル解析プログデテム後藤,《インターフェース》1978年8月号 PP.51～55
〔20〕 リスト処理と言语解析
藤野，朝倉書店，PP.5～14



封面设计：薛太忠

统一书号：15290·421　　定　价：5.50 元